이학박사 학위논문

Polynomial Factorization and Its Applications

(다항식 인수분해와 그 응용)

2013년 2월

서울대학교 대학원

수리과학부

이형태

Polynomial Factorization and Its Applications

(다항식 인수분해와 그 응용)

지도교수 천정희

이 논문을 이학박사 학위논문으로 제출함

2012년 11월

서울대학교 대학원

수리과학부

이형태

이형태의 이학박사 학위논문을 인준함

2012년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Polynomial Factorization and Its Applications

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

to the faculty of the Graduate School of

Seoul National University

by

Hyung Tae Lee

Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Science

Seoul National University

February 2013

Abstract

Polynomial Factorization and Its Applications

Hyung Tae Lee

Department of Mathematical Sciences

The Graduate School

Seoul National University

Polynomial representations which represent a set by a polynomial over a ring $\mathbb{Z}_\sigma$ for a composite integer σ enable us to construct efficient private set operation protocols by combining with some additive homomorphic encryption scheme. However, a polynomial representation has a limitation due to the hardness of polynomial factorizations over $\mathbb{Z}_\sigma$. It makes hard to recover a corresponding set from a resulting polynomial in some private set operation protocols.

We provide two representations of a set by a polynomial over $\mathbb{Z}_\sigma$ which enable us to uniquely factorize a polynomial satisfying some criteria. The first suggestion works on a ring $\mathbb{Z}_N$ for RSA modulus N, a message space of Paillier encryption. To do this, we mediate between sizes of modulus N and elements in the domain so that each element in the domain is to be a small root of a certain polynomial over $\mathbb{Z}_\sigma$ and apply Coppersmith small root finding algorithm.

In case of our second suggestion, it works on a ring $\mathbb{Z}_\sigma$ for a product σ of small primes, a message space of Naccache-Stern encryption. While the factorization of N is secret in Paillier encryption, the factorization of σ is

public. Hence, we can obtain many candidates of roots of polynomial using a polynomial factorization algorithm working on prime fields and Chinese remainder theorem. In our suggestion, to remove irrelevant candidates, we adopt a special encoding function which supports early abort strategy. As a result, we can efficiently recover a corresponding set from a polynomial in $\mathbb{Z}_\sigma[x]$ whose roots locates in the image of our encoding function.

As applications of our polynomial representations, we obtain a constant-round private set union protocols. Our construction improves the complexity than the previous best result without an honest majority assumption. We also consider a private set intersection protocol in storage model, in which the owners of sets and the recipients are separated.

Key words: Polynomial Factorization, Private Set Operation, Additive Homomorphic Encryption, Storage Model, Set Union

Student Number: No. 2008-30086

Contents

Chapter 1

Introduction

Private set operations are to compute the pre-determined set operations among players' datasets without revealing any other information. They have widely applications, for example, to determine do-not-fly list with private set intersection protocol, to distinguish suspects of an insurance fraud with private set union and intersection protocols, and private distributed network monitoring with over-threshold set union protocols. There are many researches to construct private set operation protocols for various set operations based on cryptographic tools such as polynomial representations, pseudorandom functions, blind RSA signatures, prime representations, just to name a few. Among these techniques, polynomial representations enable us to construct efficient multi-party private set operation protocols for various set operations, combining with additive homomorphic encryption schemes or secret sharing techniques.

Let us briefly explain polynomial representation of a set. Let R be a commutative ring with unity and S be a dataset which is a subset of R. In polynomial representation, a set S is represented by a polynomial f_S defined

over R whose all roots are elements in S. That is, $f_S(x) = \prod_{s_i \in S}(x - s_i) \in R[x]$. This representation gives various set operations from polynomial operations. For instance, the sum of polynomials representing sets, represents the intersection of sets from the relation $f_{S_1} + f_{S_2} + \cdots + f_{S_n} = \gcd(f_{S_1}, f_{S_2}, \cdots, f_{S_n}) \cdot u$ for some polynomial u. The multiplication of polynomials representing sets, represents the union of sets as multi-set from the relation $f_{S_1} \cdot f_{S_2} \cdot \cdots \cdot f_{S_n} = \prod_{s_i \in \bigcup_{j=1}^{n} S_j} (x - s_i)$. Based on these facts, Kissner and Song [KS05] proposed efficient multi-party set operation protocols including set intersection, (over-threshold) multi-set union, element reduction and so on, by combining with an additive homomorphic encryption scheme and other cryptographic tools.

In private set operation protocols based on polynomial representation, the final phase is to recover the resulting set from the resulting polynomial. In case of set intersection protocol, each player has a dataset which contains the resulting set and all elements in the intersection are roots of the resulting polynomial. Hence he can obtain the intersection from the resulting polynomial by evaluating the resulting polynomial at all elements in his dataset. In case of other set operations such as set union, however, each player does not have any datasets containing the resulting set except the universe and hence it is hard to efficiently obtain the result by the same method for the case of the set intersection protocol since the size of the universe is much larger than the size of resulting set in general. As an alternative to resolve this problem, one can consider to exploit polynomial factorization algorithm or small root finding algorithm for finding all roots of the resulting polynomial.

Let us focus on polynomial factorization of a polynomial defined over R which is a message space of an additive homomorphic encryption scheme. As far as we know, there are three types of message spaces of additive homomorphic encryption schemes: (1) $\mathbb{Z}_p$ with hidden prime p in Okamoto-

Uchiyama encryption [OU98], (2) $\mathbb{Z}_N$ with RSA modulus N in Paillier encryption [Pai99] or Camenisch-Shoup encryption [CS03], and (3) $\mathbb{Z}_\sigma$ with public σ which is a product of small primes in Naccache-Stern encryption [NS98]. In the first case, however, it is impossible to factorize a polynomial in $R[x]$ since the order p is hidden information. In other cases $R[x]$ is not a unique factorization domain and hence it does not support the unique polynomial factorization in $R[x]$. Therefore, there is no efficient additive encryption scheme whose message space supports efficient polynomial factorization. (One can think that additive ElGamal [ElG84] encryption scheme is suitable since its message space is $\mathbb{Z}_p$ for some public prime p. However, it suffers from heavy decryption cost because it has to solve a discrete logarithm problem on a group of order p.)

To overcome this obstacle, previous protocols devised other methods which can recover a set at the final phase. Some private set union protocols [KS05, Fri07] exploit a mix-net protocol [KS05, FS01] instead of polynomial factorization, which runs in linear of the number of corrupted players with heavy communication and computation cost. Other protocols [SCK12] hire secret sharing techniques so that R is to be $\mathbb{Z}_p$ for some public prime p. However, it requires an honest majority assumption for security.

1.1 Contribution of This Work

The contribution of this dissertation consists of twofold: (1) two polynomial representations supporting to uniquely factorize a polynomial satisfying some criteria, defined over a message space of some additive homomorphic encryption schemes and (2) its applications to private set operation protocols.

The second part also divides into twofold, to improve private set union

protocols and to give the scenario of private set intersection in storage model and present concrete construction.

1.1.1 Our Polynomial Representation

We provide two polynomial representations that enable us to uniquely factorize a polynomial satisfying certain criteria, defined over a message space of Paillier encryption scheme or Naccache-Stern encryption scheme.

First, we focus that small roots of a polynomial over $\mathbb{Z}_N$ can be found in polynomial time in the degree of a polynomial and the size of modulus N. In [Cop97], Coppersmith provided an algorithm to find small roots of a polynomial over $\mathbb{Z}_N$ using lattice theory and he applied his algorithm to attack RSA encryption scheme for a small public key. In our first polynomial representation, we set the RSA modulus N large enough so that all elements of players' dataset are relatively small and hence we can recover a set from a polynomial using Coppersmith algorithm.

Second, we focus that the factorization of σ is public when a message space of Naccache-Stern encryption is $\mathbb{Z}_\sigma$. For a given polynomial $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$, since the factorization of $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$ for primes q_j's is public, we can compute all roots of the polynomial f over $\mathbb{Z}_{q_j}$ for all j by using a polynomial factorization algorithm over a finite field. Then we can obtain all the candidates of root of f using Chinese remainder theorem, but the number of candidates is $d^{\bar{\ell}}$, which is exponential in the size of the universe.

To efficiently remove irrelevant elements among candidates, we introduce our new encoding function ι. Let $f = \prod_{i=1}^{d}(x - \iota(s_i)) \in R[x]$ be a polynomial encoded by our function ι. Then our encoding function aborts most irrelevant candidates without $d^{\bar{\ell}}$ Chinese remainder theorem computation, by

giving a certain relation among roots of f in $\mathbb{Z}_{q_j}[x]$ and roots of f in $\mathbb{Z}_{q_{j+1}}[x]$. As a result, we can reduce the number of candidates to $3d$ on average case and efficiently recover all the roots of f with about $3d$ additional hash computations and negligible failure probability on the security parameter if they are in the image of ι.

1.1.2 Applications to Private Set Operations

Our polynomial representation can be applied to private set operation protocols. Because the recoverable size of roots is in inverse proportion to the degree of the resulting polynomial, our first polynomial representation is not suitable for the set union protocol. However, our second polynomial representation is suitable for the set union protocol combining with threshold Naccache-Stern additive homomorphic encryption scheme. To construct a private set union protocol, we adopt rational representation presented by Seo et al. [SCK12], by transforming our polynomial representation into rational function using reversed Laurent series. And we achieve to obtain the first constant-round private set union protocol without an honest majority assumption, combining with Naccache-Stern encryption scheme, instead of exploiting secret sharing techniques. Consequently, our protocols in the honest-but-curious model have comparable complexity with the previous best result [SCK12] and those in malicious model are more efficient than the previous best result [SCK12] without an honest majority assumption.

Remark that we can easily extend our set union protocol to a multi-set union protocol by modifying our encoding function so that the same elements correspond to different elements. The resulting multi-set union protocol is a little bit slower than the previous best result [HKK$^+$11]. However, the public key size of the utilized additive homomorphic encryption in our protocol is

$O(1)$, while that of previous work is $O(d)$ for the size d of the multi-set union.

We also consider a private set intersection protocol for multiple uses in storage model which is as follows: As cloud service develops, the owners of sets store their datasets in storage as encrypted. When the recipients who may not possess any datasets except the universe, want to compute an intersection of a part of stored sets in the storage, the storage provides ciphertexts corresponded to the set intersection to the recipients. Then, the recipients obtain the set intersection by jointly decrypting given ciphertexts. This model is suitable for the cases where the data owners and the recipients should be separated such as privacy-preserving data publishing [FKWY10].

There are two big differences between set intersection protocol in basic model and storage model. First, since the data owners and the data receivers are separated in storage model, the data receivers do not have any datasets except the universe. Hence they should be able to recover the result of the intersecting the sets in other ways. Second, the encrypted sets stored in storage should be able to be reused without leaking any other information.

To make up for the first difference we utilize our first polynomial representation, so that each recipient can recover the set intersection from the resulting polynomial. In this case, the cardinality of set intersection causes a problem for recovering the correct set from the resulting polynomial since it is related to the degree of the resulting polynomial and hence the possible size of roots which can recover using Coppersmith algorithm also depends on the degree of the resulting polynomial. To overcome this problem, we borrow the bucketing technique from [FNP04] and all the elements in the domain are distributed into a number of buckets using collision-resistant uniform hash function. Then the set owner represents his set as a polynomial per each bucket, not as a whole set. Thus, a set is represented by a number of polyno-

mials whose degrees are small and the result of the protocol is also a number of polynomials whose degrees are small, instead one polynomial with large degree.

For the second difference, we employ a slightly modified version of the threshold Paillier encryption scheme. In original threshold Paillier encryption, the decryption entities share the decryption key at the setup algorithm: each entity i has a secret share sk_i such that $\sum_{i=1}^{m} \mathsf{sk}_i = c\phi(N)$ for RSA modulus N, a hidden integer c, and the number of the decryption entities m. In modified version, the key generation algorithm distributes t different Pailler decryption shares $\mathsf{sk}_{1,i}, \cdots, \mathsf{sk}_{t,i}$ to each recipient i such that $\sum_{i=1}^{m} \mathsf{sk}_{j,i} = c_j\phi(N)$ for $j = 1, \ldots, t$ where c_j's are hidden integers. This method makes it possible for the recipients to perform private set intersection protocols with stored encrypted sets by t times. We also provide a way to get only $O(t/m)$ decryption shares to enable t computations. As a result, we obtain a private set intersection protocol for t-time uses in storage model.

1.2 Organization

Chapter 2 reviews a number of cryptographic concepts which are exploited in our polynomial representation and private set operations. Chapter 3 provides our two polynomial representations that enable us to recover a set from the resulting polynomial in private set operation protocols based on polynomial representation. Private set union based on our polynomial representation is given in Chapter 4 and private set intersection in storage model is presented in Chapter 5.

Chapter 2

Preliminaries

In this chapter, we briefly look into some definitions and related work of private set operations. Then we provide previous polynomial representation of a set and its variants for private set operations. We also look into why polynomial factorization over message spaces of the existing additive homomorphic encryption schemes are hard. We conclude this chapter with some factorization algorithms, which exploit in our protocol as auxiliary tools.

2.1 Basics

Throughout this dissertation, we denote the number of players by n and the cardinality of player's dataset by k in private set operation protocols. Also, d denotes the cardinality of union among datasets of players in set union protocol.

Let $\mathbb{Z}_q[x]$ be a set of polynomials defined over $\mathbb{Z}_q$ and $\mathbb{Z}_q(x)$ be a set of rational functions defined over $\mathbb{Z}_q$ for some positive integer q, i.e., $\mathbb{Z}_q(x) := \{\frac{f}{g} | f, g \in \mathbb{Z}_q[x] \text{ and } \gcd(f, g) = 1\}$. For a polynomial f defined over a commutative ring R, we denote a coefficient of x^i in a polynomial f by $f[i]$,

i.e., $f(x) = \sum_{i=0}^{\deg f} f[i]x^i \in R[x]$. For a polynomial $\sum_{i=0}^{\deg f} f[i]x^i \in Z_\sigma[x]$ and a factor q of σ, $f \bmod q$ denotes a polynomial $\sum_{i=0}^{\deg f}(f[i] \bmod q)x^i \in Z_\sigma[x]$.

A function $g : \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $\mu(\lambda)$, there exists an integer N such that $g(\lambda) < 1/\mu(\lambda)$ for all $\lambda > N$.

2.2 Private Set Operations

Problem Definition

There are n players, $\mathcal{P}_1, \ldots, \mathcal{P}_n$; each player $\mathcal{P}_i$ has a private dataset $S_i \subseteq \mathcal{U}$ of size k in the known universe $\mathcal{U}$. We define a *private set operation* problem as follows: all players learn the resulting set operation among $S_1, \ldots, S_n$ of players' datasets without revealing any other information.

We extend the definition of private set operation protocol into the storage model, in which the set owners and the recipients of the intended set may be different. In the storage model, there exist players $\mathcal{P}_1, \ldots, \mathcal{P}_n$ who have private datasets $S_i \subseteq \mathcal{U}$ of size k, respectively, and decryptors $\mathcal{D}_1, \ldots, \mathcal{D}_m$ who want to obtain the result of set operations among datasets S_i's , but does not have any datasets except the universe. In this model, all players store set encryptions of his dataset in the storage and if collusions among all decryptors are not allowed, all decryptors do not learn any information other than the resulting sets although stored sets are multiple-time utilized.

Security Model

In case of private set operation protocol, there exist two adversary models, honest-but-curious adversaries and malicious adversaries. Let us give a brief introduction on these adversary models. (See [Gol04] for formal definitions of these adversary models.) Basically, the security of protocol in the storage

model also follows these security models.

Honest-But-Curious Adversaries In this model, all players follow the described actions in the protocol and may try to gain additional information other than the result of the protocol from transcripts produced during executing the protocol. To prove the security against honest-but-curious adversaries, we show that no player learns additional information other than the result in the ideal model in which each player sends his own dataset to a trusted third party (TTP) and receives the output from the TTP.

Malicious Adversaries In this model, it allows an adversary to perform arbitrary actions including deviating from the protocol in order to gain additional information other than the result of the protocol or disturb that players receive the correct result of the protocol. To prove the security against malicious adversaries, we show that for any adversary executing a successful attack in the real protocol, there exists a simulator who executes the same attack in the ideal world.

History of Private Set Operations

There have been some researches to construct private set operation protocols [GMW87, BOGW88] with general MPC techniques. Freedman et al. [FNP04] first introduced the problem to construct efficient private set intersection protocols and gave solutions to construct 2-party set intersection protocol based on polynomial representation. Kissner and Song [KS05] improved Freedman et al.'s techniques and gave multi-party private set operation protocols for various set operations, including set intersection, multi-set union, element reduction, and so on. Later, Sang and Shen [SS09] improved Kissner and Song's protocols by developing zero-knowledge proof techniques using Boneh-Goh-Nissim encryption scheme [BGN05].

There have been some following researches to construct more efficient private set operation protocols with various techniques. In case of set intersection, Jarecki and Liu [JL09] proposed 2-party protocols based on pseudorandom functions. De Cristofaro and Tsudik [CT10] and De Cristofaro et al. [CKT10] presented 2-party protocols based on blind RSA signatures in the honest-but-curious model and malicious model, respectively. Kim et al. [KLC12] constructed 2-party protocols based on prime representation in the honest-but-curious case to reduce the communication complexity.

In case of set union, Frikken [Fri07] and Sand and Shen [SS09] presented more efficient multi-party set union protocols and multi-set union protocols in the malicious case, respectively. However, these protocols exploit a mix-net protocol [FS01] instead of polynomial factorization algorithm, which runs in linear of the number of corrupted players, and hence it cannot have constant round complexity. Recently, Seo et al. [SCK12] proposed constant round multi-party set union protocols by representing elements in a set as poles of a rational function. But, their constructions hire a secret sharing technique for supporting privacy-preserving multiplications, thus require an honest majority assumption for security and computational and communicational complexity heavier than previous results.

In case of multi-set union protocols, Hong et al. [HKK+11] proposed a protocol based on El Gamal encryption schemes defined over an extension field $\mathbb{F}_{q^\kappa}$ where κ is larger than nk where n is the number of players and k is the size of datasets. But it suffers from the public key size of the utilized encryption, since the extension degree κ of extension field has to be larger than d for the size d multi-set union and hence the public key size is $O(d)$.

2.3 Polynomial Representation

Now, we provide a polynomial representation and its variant utilized in previous private set operation protocols. Let R be a commutative ring with unity and S be a subset of R.

Polynomial Representation

In polynomial representation, a set S can be represented by a polynomial $f_S(x) \in R[x]$ whose roots are the elements of S. In other words,

$$f_S(x) := \prod_{s_j \in S} (x - s_j).$$

Then, this representation gives the following relation:

$$f_S(x) + f_{S'}(x) = \gcd(f_S(x), f_{S'}(x)) \cdot u(x)$$

for some polynomial $u(x) \in R[x]$. Hence the roots of a polynomial $f_S(x) + f_{S'}(x)$ are the elements of $S \cap S'$ with overwhelming probability. Also, the roots of $f_S(x) \cdot f_{S'}(x)$ are the elements of $S \cup S'$ as multi-sets.

Rational Representation

Recently, Seo et al. [SCK12] introduced a novel representation of a set $S \subset R$ by a rational function F_S over R whose poles consist of the elements of S. In other words,

$$F_S(x) := \frac{1}{\prod_{s_i \in S}(x - s_i)} = \frac{1}{f_S(x)}.$$

Then this representation gives the following relation:

$$F_S(x) + F_{S'}(x) = \frac{f_S(x) + f_{S'}(x)}{f_S(x) \cdot f_{S'}(x)} = \frac{\gcd(f_S(x), f_{S'}(x)) \cdot u(x)}{f_S(x) \cdot f_{S'}(x)} = \frac{u(x)}{\mathrm{lcm}(f_S(x), f_{S'}(x))}$$

for some polynomial $u(x) \in R[x]$ which is relatively prime to $\mathrm{lcm}(f_S(x), f_{S'}(x))$. Hence, if $\gcd(\mathrm{lcm}(f_S(x), f_{S'}(x)), u(x)) = 1$, then the denominator of $F_S(x) +$

$F_{S'}(x)$ has exactly the roots of $\mathrm{lcm}(f_S(x), f_{S'}(x))$, which are the elements of $S \cup S'$ as sets, not multi-sets. This rational function representation can be realized by using an infinite formal power series, so called a *reversed Laurent series*, in [SCK12].

2.4 Additive Homomorphic Encryption

Let us consider a commutative ring R with unity and a R-module G where $r \cdot g := g^r$ for $r \in R$ and $g \in G$. Let $\mathsf{E}_{\mathsf{pk}} : R \to G$ be a public key encryption under the public key pk. We can define a public key encryption for a polynomial $f = \sum_{i=0}^{\deg f} f[i]x^i \in R[x]$ as follows:

$$\mathcal{E}_{\mathsf{pk}}(f) := \sum_{i=0}^{\deg f} \mathsf{E}_{\mathsf{pk}}(f[i])x^i.$$

Assume E_{pk} is an additive homomorphic encryption scheme such that

$$\mathsf{E}_{\mathsf{pk}}(a+b) = \mathsf{E}_{\mathsf{pk}}(a)\mathsf{E}_{\mathsf{pk}}(b), \quad \mathsf{E}_{\mathsf{pk}}(ab) = \mathsf{E}_{\mathsf{pk}}(a)^b$$

for $a, b \in R$. Then given two polynomials $f = \sum_{i=0}^{\deg f} f[i]x^i$ and $g = \sum_{i=0}^{\deg g} g[i]x^i$, we can induce homomorphic properties of $\mathcal{E}$ as follows:

- Polynomial addition: Given $\mathcal{E}_{\mathsf{pk}}(f)$ and $\mathcal{E}_{\mathsf{pk}}(g)$, it is possible to compute $\mathcal{E}_{\mathsf{pk}}(f+g)$ by calculating $\mathsf{E}_{\mathsf{pk}}((f+g)[i]) = \mathsf{E}_{\mathsf{pk}}(f[i])\mathsf{E}_{\mathsf{pk}}(g[i])$ for all $0 \le i \le \max\{\deg f, \deg g\}$ where $f+g = \sum_{i=0}^{\max\{\deg f, \deg g\}} (f+g)[i]x^i$.
- Polynomial multiplication: Given $\mathcal{E}_{\mathsf{pk}}(f)$ and g, it is possible to compute $\mathcal{E}_{\mathsf{pk}}(fg)$ by calculating $\mathsf{E}_{\mathsf{pk}}((fg)[\ell]) = \prod_{i+j=\ell} \mathsf{E}_{\mathsf{pk}}(f[i])^{g[j]}$ for all $0 \le \ell \le \deg f + \deg g$ where $fg = \sum_{\ell=0}^{\deg f + \deg g} (fg)[\ell]x^\ell$.

There are several efficient additive homomorphic encryption schemes: Under the assumption that factoring $N = p^2q$ is hard, Okamoto and Uchiyama

proposed a scheme with $R = \mathbb{Z}_p$ and $G = \mathbb{Z}_N$, in which the order p of the message space R is hidden [OU98]. With the decisional composite residuosity assumption, Paillier scheme [Pai99] and Camenisch and Shoup scheme [CS03] have $R = \mathbb{Z}_N$ and $G = \mathbb{Z}_{N^2}$ for $N = pq$, in which the size of message space is a hard-to-factor composite integer N. Naccache and Stern [NS98] proposed a scheme with $R = \mathbb{Z}_\sigma$ and $G = \mathbb{Z}_N$ under the higher residue assumption, where $N = pq$ is a hard-to-factor integer and σ is a product of small primes dividing $\phi(N)$. Note that one may obtain an additive homomorphic encryption scheme from modifying El Gamal encryption [ElG84]. However, we exclude this scheme because it has to solve a discrete logarithm algorithm for a decryption.

In the first scheme, it is hard to find the roots of a polynomial in $R[x]$ without knowing a secret key since the order p of the message space R is a secret key. In the second scheme, it also looks hard to find roots of a polynomial in $R[x]$ since the factorization of N is secret information. However, it is known that if the roots of a polynomial in $R[x]$ is small enough, one can find these small roots without secret information by Coppersmith small root finding algorithm [Cop97]. Hence, if we raise the bit size of modulus with respect to the bit size of each element, we can obtain polynomial representation which can recover the set from the polynomial and one of our suggestions is based on this idea.

While, in the Naccache-Stern scheme, it may be possible to compute some roots of a polynomial in $\mathbb{Z}_\sigma[x]$ since the factorization of σ is public. But $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain and so the number of roots of $f \in \mathbb{Z}_\sigma[x]$ can be larger than the $\deg f$. In fact, if $f(x) = \prod_{i=1}^{d}(x - s_i) \in R[x]$, then the number of candidates of roots of f is $d^{\bar{\ell}}$ where $\bar{\ell}$ is the number of prime factors of σ. Our another polynomial representation uses the Naccache-

Stern scheme by introducing a method to efficiently recover all the roots of $f \in \mathbb{Z}_\sigma[x]$ satisfying some criterion.

2.4.1 Threshold Additive Homomorphic Encryption

Most private set operation protocols based on polynomial representation, including our protocols, exploit threshold additive homomorphic encryption schemes. In this subsection, we present threshold version of Pailler encryption scheme and Naccache-Stern encryption scheme which go with our polynomial representation. In case of Pailler encryption scheme, to the best of our knowledge, there have been two threshold versions [FPS01, CDN01]. We will briefly give one scheme of threshold Pailler encryption scheme.

However, as far as we know, there is no threshold version of Naccache-Stern encryption. But, we can easily construct a threshold version of Naccache-Stern additive homomorphic encryption scheme based on transforming the original Paillier homomorphic encryption scheme into a threshold version working from Shoup's technique [Sho00]. We will also briefly present a threshold version of Naccache-Stern encryption scheme.

Threshold Pailler Encryption

We briefly describe a threshold version of Paillier encryption scheme proposed in [FPS01]. The scheme is the following:

- $\mathsf{KeyGen}(\lambda, n, t)$: this algorithm is executed by a dealer. It takes as an input a security parameter λ, the number n of participating users and the threshold parameter t. Then the dealer runs the following steps:

  1. Choose an integer N, a product of two strong primes p and q such that $p = 2p'+1$, $q = 2q'+1$ for primes p', q' and $\gcd(N, \phi(N)) = 1$.

2. Choose randomly $(a, b) \in \mathbb{Z}_N^* \times \mathbb{Z}_N^*$ and let $g = (1+N)^a b^N \bmod N^2$.

3. Choose a random element β in $\mathbb{Z}_N^*$. The secret key $\beta p'q'$ is shared with the Shamir's secret sharing scheme [Sha79]: Let $f(x) = \sum_{i=0}^{t} a_i x^i$ where $a_0 = \beta p'q'$ and a_i's are randomly chosen element in $\mathbb{Z}_{Np'q'}$. Then the secret share sk_i for a user $\mathcal{P}_i$ is $f(i) \bmod N\phi(N)$.

4. Let $\theta = L(g^{p'q'\beta}) = ap'q'\beta \bmod N$ where $L(u) = \dfrac{u-1}{N}$.

5. Choose vk randomly which generates of the cyclic group of squares in $\mathbb{Z}_{N^2}^*$ and let $\mathsf{vk}_i = \mathsf{vk}^{\Delta \mathsf{sk}_i} \bmod N$ where $\Delta = n!$.

It outputs a public key $\mathsf{pk} = (N, g, \theta, \mathsf{vk}, \{\mathsf{vk}_i\}_{i=1}^n, \Delta)$ and a secret key sk_i for each user $\mathcal{P}_i$.

– $\mathsf{Enc}(\mathsf{pk}, M)$: this algorithm takes as input a public key pk and a message M. It chooses a random element x in $\mathbb{Z}_N$ and outputs $c = g^M x^N \bmod N^2$.

– $\mathsf{ShareDec}(\mathsf{pk}, i, \mathsf{sk}_i, c)$: this algorithm takes as an input a public key pk, an index $1 \le i \le n$, a secret share sk_i and a ciphertext c. Then it computes $c_i = c^{2\Delta \mathsf{sk}_i} \bmod N^2$ along with its proof π_{c_i} of the equality of the discrete logarithm between $(c^{4\Delta}, c_i^2)$ and $(\mathsf{vk}^{\Delta}, \mathsf{vk}^{\Delta \mathsf{sk}_i})$. The non-interactive version of the proof of the equality of the discrete logarithm can be easily obtained from [Sho00]. It outputs (c_i, π_{c_i}).

– $\mathsf{Combine}(\mathsf{pk}, S, c, \{c_i\}_{i\in S}, \{\pi_{c_i}\}_{i\in S})$: this algorithm is executed by the combiner. It takes as an input a public key pk, an index set S, a ciphertext c, its decryption shares $\{c_i\}_{i\in S}$, corresponding proofs $\{\pi_{c_i}\}_{i\in S}$. Then it runs the following steps:

1. If $|S| \leq t$, this algorithm returns $\perp$.
2. Otherwise, it verifies all proofs π_{c_i}'s. If at least one of proofs are failed to verify, return $\perp$.
3. For a set S, compute

$$m' = L\left(\prod_{j \in S} c_j^{2\mu_{0,j}^S} \bmod N^2\right) \times \frac{1}{4\Delta^2\theta} \bmod N$$

where $\mu_{0,j}^S = \Delta \times \prod_{j' \in S\{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$.

It outputs m'.

Threshold Naccache-Stern Encryption Scheme

Our threshold version of the Naccache-Stern encryption scheme consists of the following four algorithms:

– $\mathsf{KeyGen}(\lambda, n, t)$: this algorithm is executed by a dealer. It takes as an input a security parameter λ, the number n of participating users and the threshold parameter t. Then the dealer runs the following steps:

1. Generate $\bar{\ell}$ $(\tau(\lambda)+1)$-bit primes q_i's where $\tau(\lambda)$ is a polynomial in λ. We simply write $\tau(\lambda)$ by τ.Let $u = \prod_{i=1}^{\bar{\ell}/2} q_i$ and $v = \prod_{i=\bar{\ell}/2+1}^{\bar{\ell}} q_i$.
2. Generate two $\kappa(\lambda)$-bit large primes p_1', p_2'.
3. Compute $p_1 = 2up_1' + 1$ and $p_2 = 2vp_2' + 1$. If at least one of p_1 and p_2 are not prime, go to the first step. Otherwise, let $N = p_1p_2$ and $\sigma = uv$.
4. The secret key $\phi(N)/\sigma$ is shared with the Shamir's secret sharing scheme [Sha79]: Let $f(x) = \sum_{i=0}^{t} a_i x^i$ where $a_0 = \phi(N)/\sigma$ and a_i's are randomly chosen element in $\mathbb{Z}_{\phi(N)}$. Then the secret share sk_i for a user $\mathcal{P}_i$ is $f(i) \bmod \phi(N)/\sigma$.

5. Choose a random element g of order $\phi(N)/4$ in $\mathbb{Z}_N^*$.
6. Choose vk randomly which generates of the cyclic group of squares in $\mathbb{Z}_N^*$ and let $\mathsf{vk}_i = \mathsf{vk}^{\Delta \mathsf{sk}_i} \bmod N$ where $\Delta = n!$.

It outputs a public key $\mathsf{pk} = (N, g, \sigma, \mathsf{vk}, \{\mathsf{vk}_i\}_{i=1}^n, \Delta)$ and a secret key sk_i for each user $\mathcal{P}_i$.

– $\mathsf{Enc}(\mathsf{pk}, m)$: this algorithm takes as input a public key pk and a message m. It chooses a random element x in $\mathbb{Z}_N$ and outputs $c = g^m x^\sigma \bmod N$.

– $\mathsf{ShareDec}(\mathsf{pk}, i, \mathsf{sk}_i, c)$: this algorithm takes as an input a public key pk, an index $1 \leq i \leq n$, a secret share sk_i and a ciphertext c. Then it computes $c_i = c^{2\Delta \mathsf{sk}_i}$ along with its proof π_{c_i} of the equality of the discrete logarithm between (c^4, c_i^2) and $(\mathsf{vk}, \mathsf{vk}^{\Delta \mathsf{sk}_i})$ and computes $g_i = g^{\Delta \mathsf{sk}_i}$ along with its proof π_{g_i} of the equality of the discrete logarithm between $(g, g^{\Delta \mathsf{sk}_i})$ and $(\mathsf{vk}, \mathsf{vk}^{\Delta \mathsf{sk}_i})$. The non-interactive version of the proof of the equality of the discrete logarithm can be easily obtained from [Sho00]. It outputs $(c_i, \pi_{c_i}, g_i, \pi_{g_i})$.

– $\mathsf{Combine}(\mathsf{pk}, S, c, \{c_i\}_{i\in S}, \{\pi_{c_i}\}_{i\in S}, \{g_i\}_{i\in S}, \{\pi_{g_i}\}_{i\in S})$: this algorithm is executed by the combiner. It takes as an input a public key pk, an index set S, a ciphertext c, its decryption shares $\{c_i\}_{i\in S}$, corresponding proofs $\{\pi_{c_i}\}_{i\in S}$, generator shares $\{g_i\}_{i\in S}$, and corresponding proofs $\{\pi_{g_i}\}_{i\in S}$. Then it runs the following steps:

1. If $|S| \leq t$, this algorithm returns $\perp$.
2. Otherwise, it verifies all proofs π_{c_i}'s and π_{g_i}'s. If at least one of proofs are failed to verify, return $\perp$.
3. For a set S, compute $\mu_{0,j}^S := \Delta \prod_{j' \in S\setminus\{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$. Then compute $\bar{c} = \prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod N$ and $\bar{g} = \prod_{j\in S} g_j^{2\mu_{0,j}^S} \bmod N$

4. For all $q_i|\sigma$, compute $m'_i = \log_{\bar{g}^{\hat{q}_i}} \bar{c}^{\hat{q}_i}$ where $\hat{q}_i = \frac{\sigma}{q_i}$. Then compute m' such that $m' \equiv m'_i \bmod q_i$ for all i using CRT. It outputs m'.

Correctness of Our Construction By the Lagrange interpolation formula, we obtain

$$\bar{c} = \prod_{j \in S} c_j^{2\mu_{0,j}^S} = \prod_{j \in S} c^{4\Delta \mathsf{sk}_i \mu_{0,j}^S} = c^{4\Delta \sum_{j \in S} \mathsf{sk}_j \mu_{0,j}^S} = c^{4\Delta^2 f(0)}$$

and similarly $\bar{g} = g^{4\Delta^2 f(0)}$. Hence,

$$\begin{aligned} \log_{\bar{g}^{\hat{q}_i}} \bar{c}^{\hat{q}_i} &= \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} c^{4\Delta^2 \frac{\phi(N)}{q_i}} = \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} \left(g^m x^\sigma\right)^{4\Delta^2 \frac{\phi(N)}{q_i}} \\ &= \log_{g^{4\Delta^2 \frac{\phi(N)}{q_i}}} \left(g^m\right)^{4\Delta^2 \frac{\phi(N)}{q_i}} \equiv (m \bmod q_i), \end{aligned}$$

it gives the correctness of our construction.

The security proof of this scheme is almost similar with that of a threshold version of Paillier encryption scheme. We omit the details.

2.5 Polynomial Factorization Algorithm

When R is a finite field $\mathbb{F}_q$, we have several efficient root finding algorithms in $R[x]$. Using a square-free decomposition and the Cantor-Zassenhaus algorithm [vzGG99, Section 14.4], a polynomial of degree d over a field $\mathbb{F}_q$ is factored in $\widetilde{O}(d^2 \log q)$ field operations. Recently, it has been improved using fast arithmetic into $O(d^{1.5+o(1)})$ field operations by Umans [Uma08]. However, as mentioned above, there is no efficient additive homomorphic encryption whose message space is a finite field $\mathbb{F}_q$ with public q.

However, there is no polynomial factorization algorithm defined over $\mathbb{Z}_N$. Moreover, when R is $\mathbb{Z}_N$ for RSA modulus N, Shamir [Sha93] showed that to factor a polynomial which has only linear factors in $\mathbb{Z}_N[x]$ is equivalent to factor N.

Consider $R = \mathbb{Z}_\sigma$, a message space of Naccache-Stern encryption scheme for $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$ with distinct primes q_j's. Let $f \in \mathbb{Z}_\sigma[x]$ be a polynomial of degree d, which is a product of d linear factors $(x - s_i)$'s. Since $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain, it is still very hard to recover the exact $(x - s_i)$'s from the polynomial f.

Chapter 3

Polynomial Factorization over $\mathbb{Z}_\sigma$ for composite σ

There are three types of message spaces R of efficient additive homomorphic encryption schemes: (1) $R = \mathbb{Z}_p$ with a hidden prime p [OU98], (2) $R = \mathbb{Z}_N$ for an RSA modulus N [Pai99, CS03], and (3) $R = \mathbb{Z}_\sigma$ for a public product σ of primes [NS98]. Consider a polynomial f defined over a message space R of an efficient additive homomorphic encryption scheme. In the first case, since the order of the message space R is hidden, there is no method to factorize a polynomial defined over R. Also, in the other cases, since $R[x]$ is not a unique factorization domain, it is impossible to uniquely factorize a polynomial defined over R. Hence, these facts prevent us from recovering a correct set from a polynomial in polynomial representation.

We provide our two polynomial representations that enable us to uniquely factorize a polynomial defined over message spaces of some additive homomorphic encryption schemes when the polynomial satisfies some criteria.

Our first suggestion works on the message space of Paillier additive homomorphic encryption scheme. To do this, we mediate the size of the modulus

of Paillier encryption and the universe and then apply small root finding algorithm over $\mathbb{Z}_N$, originally proposed for the purpose of attacking RSA cryptosystem with small public key by Coppersmith [Cop97].

Our second suggestion works on the message space of Naccache-Stern additive homomorphic encryption scheme. We focus on the fact that the factorization of σ is public when $\mathbb{Z}_\sigma$ is the message space of Naccache-Stern encryption. For a given polynomial $f(x) = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$, we can obtain all roots of the polynomial over $\mathbb{Z}_{q_j}$ when $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$. Then we can obtain $d^{\bar{\ell}}$ candidates of roots using CRT computation, however we cannot determine the exact roots s_i's among candidates. To resolve this issues, we introduce our special encoding function and give some relation between roots of f in $\mathbb{Z}_{q_j}[x]$ and roots of $\mathbb{Z}_{q_{j+1}}[x]$.

3.1 When the Factorization of σ is Hidden

We provide a polynomial representation which works on $\mathbb{Z}_N$, a message space of Paillier additive homomorphic encryption scheme.

Encoding Phase

Let $\mathcal{U} \subseteq \{0,1\}^u$ be the universe of datasets and S be a subset of the universe with cardinality k. Set N to be a product of two primes p, q satisfying $N \geq (2 \cdot 2^u)^{2.18k}$. Then, we represent a set S by a polynomial f_S whose all roots are elements in S . In other words,

$$f_S := \prod_{s_j \in S} (x - s_j) \in \mathbb{Z}_N[x]$$

as an usual polynomial representation.

Decoding Phase

Let us explain a method to recover a set from a polynomial representing a set using Coppersmith algorithm. For a given polynomial $f_S = \sum_{i=0}^{k} f_S[i]x^i$, one generates a $(k+1) \times (k+1)$ matrix

$$F = \begin{pmatrix} N & 0 & 0 & \cdots & 0 & 0 \\ 0 & NX & 0 & \cdots & 0 & 0 \\ 0 & 0 & NX^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & NX^{k-1} & 0 \\ f_S[0] & f_S[1]X & f_S[2]X^2 & \cdots & f_S[k-1]X^{k-1} & X^k \end{pmatrix}$$

for an integer $X > |\mathcal{U}|$. Then if one finds a short basis $b_F = (a_0, a_1X, a_2X^2, \ldots, a_kX^k)$ generated by rows of F using lattice reduction algorithm and finds all roots of a polynomial $\sum_{i=0}^{k} a_ix^i$. Then, all roots of the polynomial $\sum_{i=0}^{k} a_ix^i$ are to be all roots of the polynomial f_S since $\sum_{i=0}^{k} a_ix^i$ is a linear combination of $\sum_{i=0}^{k} a_ix^i$ and NX^i's.

Hence the correctness and the complexity of the decoding phase are dependent on those of the utilized lattice reduction algorithm to find a short basis b_F. The following theorem guarantees that one can find a suitable short vector using Coppersmith's algorithm in our suggestion.

Theorem 3.1.1 ([Cop97])**.** *Let* $0 < \epsilon < \min\{0.18, \frac{1}{k}\}$*. Let* $F(x)$ *be a monic polynomial of degree* k *with one or more small roots* x_0 *modulo* N *such that* $|x_0| < \frac{1}{2}N^{\frac{1}{k}-\epsilon}$*. Then one can find* x_0 *in polynomial time in* $k, \frac{1}{\epsilon}$*, and* $\log N$*.*

Since we have to obtain all roots in the universe $\mathcal{U}$, we can obtain recover the correct set from a given polynomial if $N \geq (2 \cdot 2^u)^{2.18k}$ from $|\mathcal{U}| \leq 2^u < \frac{1}{2}N^{\frac{1}{k}}$.

3.2 When the Factorization of σ is Public

Let $f_S(x) \in \mathbb{Z}_\sigma[x]$ be a polynomial, all of whose roots are the elements in $S \subseteq \mathbb{Z}_\sigma$ for a composite $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$ with distinct primes q_j's. Since $\mathbb{Z}_\sigma[x]$ is not a unique factorization domain, it is impossible to recover the exact S from the polynomial f_S in almost all cases. In this section, we provide our new polynomial representation that enables us to efficiently recover the exact set from the polynomial represented by our suggestion, which works on $\mathbb{Z}_\sigma$ the message space of Naccache-Stern additive homomorphic encryption scheme. Note that some parts of this subsection were collaborated with Hyunsook Hong [Hon12].

Let us explain parameters for our polynomial representation and private set operations. First, set the bit size of the modulus N of Naccache-Stern encryption scheme for security. For given universe $\mathcal{U}$ and the maximum size of the resulting set union d, let $d_0 = \max\{d, \lceil \log N \rceil\}$ and set $\tau = \frac{1}{3}(\log d + 2 \log d_0)$. Set the parameter ℓ and α so that ℓ is the smallest positive integer such that $\mathcal{U} \subseteq \{0,1\}^{3\tau\alpha\ell}$ for some rational number $0 < \alpha < 1$ satisfying $3\alpha\tau$ and $3(1-\alpha)\tau$ are integers. Note that the proper size of α is $\frac{1}{3}$ and the detail analysis will be explained later. Then, set the proper $\bar{\ell}$ larger than ℓ, the proper size will be discussed at the last of Section 3.2.1. Then, choose $\bar{\ell}$ $(3\tau+1)$-bit primes q_j's and generate parameters of Naccache-Stern encryption scheme.

Now, we are ready to describe our new polynomial representation. Focus on the fact that the factorization of σ is public in Naccache-Stern encryption scheme. Using this fact, given a polynomial $f = \prod_{i=1}^{d}(x - s_i) \in \mathbb{Z}_\sigma[x]$ for a set $S = \{s_1, \ldots, s_d\}$, one can obtain all roots of $f \bmod q_j$ for each j, by applying Umans' polynomial factorization algorithm over a finite field $\mathbb{Z}_{q_j}$. To recover S, one can perform Chinese remainder theorem computation for

obtaining less than $d^{\bar{\ell}}$ candidates of roots of f over $\mathbb{Z}_\sigma$. In general, however, the number of roots of f over $\mathbb{Z}_\sigma$ is larger than the $\deg f$ and there is no criteria to determine the exact set S. To remove irrelevant roots which are not in S, we give some relations among all roots of polynomials $f \bmod q_j$'s by providing an encoding function.

Before describing our solution, we look into an easy way to give a relation among all roots of polynomial $f \bmod q_j$. However, it is not efficient to recover a set from a polynomial.

Encoding with a Tag

To give relations among all roots of polynomials $f \bmod q_i$'s, we may consider to insert the same value depending on the element, called a *tag*, into an element part in $\mathbb{Z}_{q_j}$'s. For example, let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ be a collision-resistant hash function for a positive rational number $0 < \alpha < 1$ such that $3\alpha\tau$ and $3(1-\alpha)\tau$ are to be integers. Parse s_i into $\bar{\ell}$ blocks $s_{i,1}, \ldots, s_{i,\bar{\ell}}$ of $(3\alpha\tau)$-bit so that $s_i = s_{i,1}||\cdots||s_{i,\bar{\ell}}$. Consider a function $\iota' : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\bar{\ell}} \to \mathbb{Z}_\sigma$, in which $\iota'(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota'(s_i) \equiv s_{i,j}||h(s_i) \bmod q_j$ for $1 \le j \le \bar{\ell}$.* Then we represent a set S as a polynomial $f_S(x) = \prod_{s_i \in S}(x - \iota'(s_i)) \in \mathbb{Z}_\sigma[x]$.

Then one can reduce the number of candidates by checking a tag $h(s_i)$ when one gets all roots over $\mathbb{Z}_{q_j}$'s. However, when a collision occurs, say $h(s_i) = h(s_j)$ with $s_i \neq s_j$, one has to check the hash value of $2^{\bar{\ell}}$ elements which are possible combinations $(s_{i,1}, s_{j,1}), \cdots, (s_{i,\bar{\ell}}, s_{j,\bar{\ell}})$. The probability†

*By notation abuse, throughout this paper if necessary, we regard a bit string as the corresponding integer via some converting function.

†Pr[At least one collision occur among d hash values $h(s_i)$'s.]
$\le 1 - \left(1 - \frac{1}{2^{3(1-\alpha)\tau}}\right)\cdots\left(1 - \frac{d-1}{2^{3(1-\alpha)\tau}}\right) \approx 1 - \exp\left(-\prod_{i=1}^{d-1}\frac{i}{2^{3(1-\alpha)\tau}}\right) \approx \frac{d^2}{2\cdot 2^{3(1-\alpha)\tau}}$.

that at least one collision occur among d elements is lower bounded by $\frac{d^2}{2^{1+3\tau(1-\alpha)}}$ which is not negligible even for small α. Moreover, the expected computation becomes $\Omega(2^{\bar{\ell}})$, which is not a polynomial in d.

3.2.1 Our Polynomial Representation

Now, we present our polynomial representation for supporting to recover a set from the polynomial over $\mathbb{Z}_\sigma$ represented by our suggestion. Take $\alpha = \frac{1}{3}$, i.e., $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$ for optimization. If $\alpha \neq \frac{1}{3}$, the expected computation is in polynomial time only when the size of the universe is restricted. Details about the proper size of α are given later.

Encoding by Repetition

Let $h : \{0,1\}^* \to \{0,1\}^{2\tau}$ and $h_j : \{0,1\}^* \to \{0,1\}^{\tau}$ be uniform hash functions for $1 \leq j \leq \ell'$. Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{\tau\ell}$ into ℓ blocks $s_{i,1}, \ldots, s_{i,\ell}$ of τ-bit so that $s_i = s_{i,1}||\cdots||s_{i,\ell}$. Let $s_{i,\ell+j} = h_j(s_i)$ for $1 \leq j \leq \ell'$ and parse $h(s_i)$ into two blocks $s_{i,\bar{\ell}+1}$ and $s_{i,\bar{\ell}+2}$ of τ-bit. Let $\bar{\ell} = \ell + \ell'$. We define our encoding function $\iota : \mathcal{U} \subseteq \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$, in which $\iota(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota(s_i) \equiv s_{i,j}||s_{i,j+1}||s_{i,j+2} \bmod q_j$ for $1 \leq j \leq \bar{\ell}$. Then a set S is represented as a polynomial $f_S(x) = \prod_{s_i \in S}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$.

$$
\begin{array}{cccc}
\overbrace{s_{1,1}||\ s_{1,2}\ ||s_{1,3}} & \overbrace{s_{2,1}||\ s_{2,2}\ ||s_{2,3}} & \cdots & \overbrace{s_{d,1}||\ s_{d,2}\ ||s_{d,3}} \pmod{q_1} \\
s_{1,2}||\ s_{1,3}\ ||s_{1,4} & s_{2,2}||\ s_{2,3}\ ||s_{2,4} & \cdots & s_{d,2}||\ s_{d,3}\ ||s_{d,4} \pmod{q_2} \\
\vdots & \vdots & \ddots & \vdots \\
s_{1,\bar{\ell}}||s_{1,\bar{\ell}+1}||s_{1,\bar{\ell}+2} & s_{2,\bar{\ell}}||s_{2,\bar{\ell}+1}||s_{2,\bar{\ell}+2} & \cdots & s_{d,\bar{\ell}}||s_{d,\bar{\ell}+1}||s_{d,\bar{\ell}+2} \pmod{q_{\bar{\ell}}}
\end{array}
$$

Figure 3.1: Our Encoding Function ι

Decoding Phase

Denote by $s_j^{(i)} := \iota(s_i) \bmod q_j$ for each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$. For $1 \leq j \leq \bar{\ell} - 1$, we define $(s_j^{(i)}, s_{j+1}^{(i')}) \in \mathbb{Z}_{q_j} \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if the last (2τ)-bit of $s_j^{(i)}$ is equal to the first (2τ)-bit of $s_{j+1}^{(i')}$, i.e, $s_{i,j+1}||s_{i,j+2} = s_{i',j+1}||s_{i',j+2}$. Inductively, we also define $(s_1^{(i_1)}, \cdots, s_{j+1}^{(i_{j+1})}) \in \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if $(s_1^{(i_1)}, \cdots, s_j^{(i_j)})$ and $(s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ are linkable pairs.

Let $\iota(s_i)$ and $\iota(s_{i'})$ be images of elements s_i and $s_{i'}$ of the function ι with $s_i \neq s_{i'}$. We can easily check the following properties:

- $\left(s_1^{(i)}, \cdots, s_{j+1}^{(i)}\right)$ is always a linkable pair.
- $\Pr\left[\left(s_j^{(i)}, s_{j+1}^{(i')}\right) \text{ is a linkable pair}\right] = \Pr\left[s_{i,j+1}||s_{i,j+2} = s_{i',j+1}||s_{i',j+2}\right] = \frac{1}{2^{2\tau}}$ if s_i and $s_{i'}$ are randomly chosen strings from $\{0,1\}^{\tau\ell}$.

$$
\begin{aligned}
s_1^{(i_1)} &= s_{i_1,1}||s_{i_1,2}||s_{s_1,3} \\
s_2^{(i_2)} &= \quad s_{i_2,2}||s_{i_2,3}||s_{s_2,4} \\
s_3^{(i_3)} &= \quad\quad s_{i_3,3}||s_{i_3,4}||s_{s_3,5} \\
&\Rightarrow \left(s_1^{(i_1)}, s_2^{(i_2)}, s_3^{(i_3)}\right) \text{ is a linkable pair.}
\end{aligned}
$$

Figure 3.2: Linkable Pairs

At decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(s_i)) \in \mathbb{Z}_\sigma[x]$ is given, we perform two phases to find the correct d roots of $f(x)$. In the first stage, one computes all the roots $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ over $\mathbb{Z}_{q_j}[x]$ for each j. For each j sequentially from 1 to $\bar{\ell} - 1$, we find all the linkable pairs among $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(d)}\}$ by checking whether the last (2τ)-bit of $s_j^{(i)}$ and the first (2τ)-bit of $s_{j+1}^{(i)}$ are the same. It can be done by d^2 comparisons or $O(d \log d)$ computations using sorting and determining algorithm.

After $\bar{\ell}-1$ steps, we obtain d' number of linkable pairs of $\bar{\ell}$-tuple, which is a candidate of roots of f and an element of a set. It includes the d elements corresponding to $\iota(s_1), \ldots, \iota(s_d)$. If d' is much larger than d, it can be a burden. However, we can show that the expected value of d' is at most $3d$ in Theorem 3.2.1 on average case.

After obtaining d' linkable pairs of $\bar{\ell}$-tuple, in the second phase, we check whether each pair belongs to the image of ι with the following equalities:

$$s_{i,\ell+j} = h_j(s_i) \text{ for all } 1 \le j \le \ell', \tag{3.2.1}$$

$$s_{i,\bar{\ell}+1} || \, s_{i,\bar{\ell}+2} = h(s_i). \tag{3.2.2}$$

The linkable pairs of $\bar{\ell}$-tuple, corresponding to $\iota(s_i)$ for some i clearly satisfies the above equations. However, for a random $\bar{\ell}$-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{\bar{\ell}}}$, the probability that it satisfies the relation (3.2.1) is about $\dfrac{1}{2^{\tau\ell'}}$ and the probability that it satisfies the relation (3.2.2) is about $\dfrac{1}{2^{2\tau}}$. Hence, the expected number of wrong $\bar{\ell}$-tuples passing both phases is less than $d \times \dfrac{1}{2^{\tau(2+\ell')}}$. It is less than $2^{-\lambda}$ for a security parameter λ if we take the parameter ℓ' to satisfy

$$\ell' > \frac{3(\lambda + \log d)}{\log d + 2\log d_0} - 2. \tag{3.2.3}$$

For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, then ℓ' is about 8.

3.2.2 The Expected Number of Linkable Pairs

In this subsection, we analyze the expected number of linkable pairs of $\bar{\ell}$-tuple when we recover a set from a polynomial of degree d, represented by our suggestion.

Remark that we may assume all elements in the set union of cardinality d are randomly chosen in the universe $\mathcal{U}$. In general, a player's element may not be random in the universe $\mathcal{U}$, but one can randomize elements efficiently

using a pseudorandom permutation. For example, block ciphers are to be candidates of practical pseudorandom permutations.

A set of κ elements $s_j^{(1)}, \cdots, s_j^{(\kappa)} \in \mathbb{Z}_{q_j}$ is called a κ-*collision* if their last 2τ-bits are the same. Since $(s_{j-1}^{(1)}, s_j^{(1)})$ is a linkable pair, κ-collision causes at least κ linkable pairs. We easily obtain the following observations, which are evidences of the expected number of linkable pairs of $\bar{\ell}$-tuple is not large.

1. The probability that at least one 2-collision occurs in $\mathbb{Z}_{q_j}$ is less than $\frac{1}{2}$ by the birthday paradox.

2. The probability that at least one κ-collision occurs for $\kappa \geq 3$ in $\mathbb{Z}_{q_j}$ is at most $\frac{1}{4d} \approx \frac{1}{2^{(2+\tau)}}$ of the probability that at least one $(\kappa-1)$-collision occurs from [STKT06, Theorem 2].

3. If κ-collision occurs in $\mathbb{Z}_{q_j}$, the number of candidates of roots of f is to be κ^2 from this collision. (Assume that κ-collision $\{s_j^{(1)}, \cdots, s_j^{(\kappa)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(\kappa)}\}$ occurs. Then $s_j^{(1)}$ can be combined with κ candidates $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(\kappa)}\}$. Hence κ^2 linkable pairs are generated.)

The expected number of 2-collision in $\mathbb{Z}_{q_j}$ for all j are roughly $\frac{\bar{\ell}}{2}$ and the expected number of κ-collision in $\mathbb{Z}_{q_j}$ for $\kappa \geq 3$ is negligible. Also, the expected number of linkable pairs of $\bar{\ell}$-tuple is to be less than $\left(\frac{\bar{\ell}}{2}\right)^2$ since the probability that a 2-collision occurs under the same position is at most $\frac{1}{\binom{d}{2}} \approx \frac{1}{2^{2\tau}}$.

Theorem 3.2.1 gives a rigorous analysis of the upper bound of the expected number of linked pairs of $\bar{\ell}$-tuple.

Theorem 3.2.1. *Let f be a polynomial which has d roots. The expected number of linkable pairs of $\bar{\ell}$-tuple is at most $3d$.*

Proof. Let E_j be the expected number of linkable pairs of j-tuple in $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_j}$ for $j \geq 2$.

Then,

$$\begin{aligned} E_2 &= \sum_{i_1,i_2 \in \{1,\cdots,d\}} 1 \cdot \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair}\right] \\ &= \sum \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair and } i_1 = i_2\right] \\ &\quad + \sum \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair and } i_1 \neq i_2\right] \\ &= d + \sum_{i_2 \in \{1,\cdots,d\} \setminus \{i_1\}} \Pr\left[s_{i_1,2} || s_{i_1,3} = s_{i_2,2} || s_{i_2,3}\right] = d\left(1 + \frac{d-1}{2^{2\tau}}\right). \end{aligned}$$

If $i_{j+1} = i_{j-1}$ and $(s_{j-1}^{(i_{j-1})}, s_j^{(i_j)})$ is a linkable pair, then $s_{i_{j+1},j+1} = s_{i_{j-1},j+1} = s_{i_j,j+1}$. Therefore,

$$\begin{aligned} &\sum_{i_1,\cdots,i_{j+1} \in \{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1} = i_{j-1} \neq i_j\right] \\ = &\sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+1} || s_{i_j,j+2} = s_{i_{j+1},j+1} || s_{i_{j+1},j+2}\right] \\ = &\sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+2} = s_{i_{j+1},j+2}\right] \\ = &\sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right] \cdot \Pr\left[s_{i_j,j+2} = s_{i_{j+1},j+2}\right] \\ = &\frac{1}{2^\tau} \cdot \sum_{i_{j+1} = i_{j-1} \neq i_j} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right] \\ \leq &\frac{1}{2^\tau} \cdot \sum_{i_1,\cdots,i_j \in \{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right] \\ = &\frac{1}{2^\tau} E_j. \end{aligned}$$

Also,

$$\begin{aligned}
& \sum_{i_1,\cdots,i_{j+1}\in\{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1}\notin\{i_{j-1},i_j\}\right] \\
= & \sum_{i_{j+1}\notin\{i_{j-1},i_j\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)}) \text{ is a linkable pair and } s_{i_j,j+1}||s_{i_j,j+2}=s_{i_{j+1},j+1}||s_{i_{j+1},j+2}\right] \\
= & \sum_{i_{j+1}\notin\{i_{j-1},i_j\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)}) \text{ is a linkable pair}\right]\cdot\Pr\left[s_{i_j,j+1}||s_{i_j,j+2}=s_{i_{j+1},j+1}||s_{i_{j+1},j+2}\right] \\
\le & \frac{d-1}{2^{2\tau}}\cdot\sum_{i_1,\cdots,i_{j+1}\in\{1,\cdots,d\}} \Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)}) \text{ is a linkable pair}\right]=\frac{d-1}{2^{2\tau}}E_j.
\end{aligned}$$

From the above equations, we obtain the recurrence formula of E_j as follows:

$$\begin{aligned}
E_{j+1} &= \sum_{i_1,\cdots,i_{j+1}\in\{1,\cdots,d\}} 1\cdot\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair}\right] \\
&= \sum\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j=i_{j+1}.\right] \\
&\quad +\sum\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j\neq i_{j+1}\right] \\
&= E_j+\sum\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j-1}=i_{j+1}\neq i_j\right] \\
&\quad +\sum\Pr\left[(s_1^{(i_1)},\cdots,s_j^{(i_j)},s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1}\notin\{i_{j-1},i_j\}\right] \\
&\le \left(1+\frac{1}{2^\tau}+\frac{d-1}{2^{2\tau}}\right)\cdot E_j,
\end{aligned}$$

by induction for $j\ge 2$. Hence, $E_{\bar\ell}\le d\left(1+\frac{1}{2^\tau}+\frac{d-1}{2^{2\tau}}\right)^{\bar\ell-1}$.

Now, we show that $\min\left\{d,\frac{\lceil\log N\rceil}{12\tau}\right\}\le\frac{d_0^{1/3}d^{2/3}}{2}$. If $d_0\ge 8d$,

$$\min\left\{d,\frac{\lceil\log N\rceil}{12\tau}\right\}\le d=d^{1/3}\cdot d^{2/3}\le\left(\frac{d_0}{8}\right)^{\frac{1}{3}}d^{2/3}=\frac{d_0^{1/3}\cdot d^{2/3}}{2}.$$

Otherwise, $d_0<8d\le 4\tau d$ since $\tau\ge 2$. Then $d_0^2\le(4\tau d)^2\le 8\tau^3d^2$ and so $d_0=d_0^{1/3}\cdot d_0^{2/3}\le 2\tau d_0^{1/3}d^{2/3}$. Hence

$$\min\left\{d,\frac{\lceil\log N\rceil}{12\tau}\right\}\le\frac{\log N}{12\tau}\le\frac{d_0}{12\tau}\le\frac{d_0^{1/3}\cdot d^{2/3}}{2},$$

if $d_0 < 8d$. Therefore,

$$\min\left\{d, \frac{\lceil \log N \rceil}{12\tau}\right\} \leq \frac{d_0^{1/3} d^{2/3}}{2} = \frac{(d_0^2 d)^{2/3}}{2d_0} \leq \frac{2^{2\tau}}{2^\tau + d}$$

because $2^\tau + d \leq 2d_0$. Finally, since $\bar{\ell} \leq \min\left\{d, \frac{\lceil \log N \rceil}{12\tau}\right\} \leq \frac{2^{2\tau}}{2^\tau + d}$[‡],

$$E_{\bar{\ell}} \leq d\left(1 + \frac{1}{2^\tau} + \frac{d-1}{2^{2\tau}}\right)^{\bar{\ell}-1} < ed < 3d$$

where $e \approx 2.718$ is the base of the natural logarithm. From these results, the upper bound of the expected number of linkable pairs of $\bar{\ell}$-tuple is $3d$. □

3.2.3 The Proper Size of α

In this subsection, we explain why the proper size of α is $\frac{1}{3}$. Let $\alpha = \frac{b}{a}$ with relatively prime a, b and suppose that ℓ and ℓ' are divided by b and let $h : \{0,1\}^* \to \{0,1\}^{3(1-\alpha)\tau}$ and $h_i : \{0,1\}^* \to \{0,1\}^{3\alpha\tau}$ be uniform hash functions for $1 \leq i \leq \ell'$. Parse a message $s_i \in \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell}$ into ℓ blocks $s_{i,1}, \cdots, s_{i,\ell}$ of $3\alpha\tau$-bit so that $s_i = s_{i,1}||\cdots||s_{i,\ell}$. Let $s_{i,\ell+j} = h_j(s_i)$ for $1 \leq j \leq \ell'$ and parse $h(s_i)$ into $(a-b)$ blocks $s_{i,\bar{\ell}+1}, \cdots, s_{i,\bar{\ell}+a-b}$ of $3\alpha\tau$-bit. Let $\bar{\ell} = \ell + \ell'$. We define our encoding function $\iota'' : \mathcal{U} \subseteq \{0,1\}^{3\alpha\tau\ell} \to \mathbb{Z}_\sigma$, in which $\iota''(s_i)$ is the unique element in $\mathbb{Z}_\sigma$ satisfying $\iota''(s_i) = s_{i,(j-1)b+1}||\cdots||s_{i,(j-1)b+a} \bmod q_j$. Then a set S is represented as a polynomial $f(x) = \prod_{s_i \in S}(x - \iota''(s_i)) \in \mathbb{Z}_\sigma[x]$.

For each message $s_i = s_{i,1}||\cdots||s_{i,\ell}$, denote by $s_j^{(i)} := \iota''(s_i) \bmod q_j$. At decoding phase, when a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota''(s_i))$ is given, one computes all the roots $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ over $\mathbb{Z}_{q_j}$. Then, for each j sequentially from 1 to $\bar{\ell} - 1$, we find all linkable pairs among $\{s_j^{(1)}, \cdots, s_j^{(d)}\}$ and $\{s_{j+1}^{(1)}, \cdots, s_{j+1}^{(d)}\}$ to find the correct d s_i's.

[‡]We will set the parameter $\bar{\ell}$ satisfies this relation for complexity of our set union protocol. Refer to Section 4.2.

We generalize the definition of the linkable pair and look into some properties of linkable elements for the all cases of α. We define $(s_j^{(i)}, s_{j+1}^{(i')}) \in \mathbb{Z}_{q_j} \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if the last $3(1-\alpha)\tau$-bit of of $s_j^{(i)}$ is equal to the first $3(1-\alpha)\tau$-bit of $s_{j+1}^{(i')}$. Inductively, we also define $(s_1^{(i_1)}, \cdots, s_{j+1}^{(i_{j+1})}) \in \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_{j+1}}$ to be a *linkable pair* if $(s_1^{(i_1)}, \cdots, s_j^{(i_j)})$ and $(s_j^{(i_j)}, s_{j+1}^{(i_{j+1})})$ are linkable pairs.

Lemma 3.2.1. *For $j = 1, \cdots, \ell - 1$,*

1. *$(s_1^{(i)}, s_2^{(i)}, \cdots, s_{j+1}^{(i)})$ is always a linkable pair.*
2. *$\Pr\left[(s_j^{(i)}, s_{j+1}^{(i')}) \text{ is a linkable pair}\right] = 1 + \dfrac{d-1}{2^{3(1-\alpha)\tau}}$*
3. *$\Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j \neq i_{j+1}\right]$*
   $\leq \left(\frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right]$

Proof. 1. It is trivial by the above definition.

2.

$$\begin{aligned}
& \Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair}\right] \\
= \; & \Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair and } i = i'\right] \\
& + \Pr\left[(s_j^{(i)}, s_j^{(i')}) \text{ is a linkable pair and } i \neq i'\right] \\
= \; & 1 + \sum_{i' \in \{1, \cdots, d\} \setminus \{i\}} \Pr\left[s_{i,jb+1} || \cdots || s_{i,jb+a-b} = s_{i',jb+1} || \cdots || s_{i',jb+a-b}\right] \\
= \; & 1 + \frac{d-1}{2^{3(1-\alpha)\tau}}.
\end{aligned}$$

3.

$$\begin{aligned}
& \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_j \neq i_{j+1}\right] \\
= & \sum_{k=1}^{j} \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}, \cdots, s_{j+1}^{(i_{j+1})}) \text{ is a linkable pair and } i_{j+1} = i_{j-k} \notin \{i_{j-k+1}, \cdots, i_j\}\right] \\
\leq & \sum_{k=1}^{j} \frac{1}{\min\{2^{3k\alpha\tau}, 2^{3(1-\alpha)\tau}\}} \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right] \\
\leq & \left(\frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \Pr\left[(s_1^{(i_1)}, \cdots, s_j^{(i_j)}) \text{ is a linkable pair}\right]
\end{aligned}$$

since $\sum_{k=1}^{j} \frac{1}{2^{3k\alpha\tau}} \leq \sum_{k=1}^{\infty} \frac{1}{2^{3k\alpha\tau}} \leq \frac{2}{2^{3k\alpha\tau}}$ and $\sum_{k=1}^{j} \frac{1}{2^{3(1-\alpha)\tau}} \leq \frac{\bar{\ell}}{2^{3(1-\alpha)\tau}} \leq \frac{d}{2^{3(1-\alpha)\tau}}$.

□

Theorem 3.2.2. *Given a polynomial $f(x) = \prod_{i=1}^{d}(x - \iota(m_i))$ for the encoding function ι, the expected number of linkable pairs of $\bar{\ell}$-tuple is at most $d\left(1 + \frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}$.*

Proof. The expected number of linkable pairs of $\bar{\ell}$-tuple is

$$\begin{aligned}
& \sum_{i_1,\cdots,i_{\bar{\ell}} \in \{1,\ldots,d\}} 1 \cdot \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair.}\right] \\
= & \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair and } i_{\bar{\ell}} = i_{\bar{\ell}-1}\right] \\
& + \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}}^{(i_{\bar{\ell}})}) \text{ is a linkable pair and } i_{\bar{\ell}} \neq i_{\bar{\ell}-1}\right] \\
\leq & \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{3(1-\alpha)\tau}}\right) \sum \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}-1}^{(i_{\bar{\ell}-1})}) \text{ is a linkable pair}\right] \\
\leq & \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{1}{2^{(2-3\alpha)\tau}}\right) \sum 1 \cdot \Pr\left[(s_1^{(i_1)}, \cdots, s_{\bar{\ell}-1}^{(i_{\bar{\ell}-1})}) \text{ is a linkable pair}\right] (\because d < 2^\tau) \\
\leq & \left(1 + \frac{2}{2^{3\alpha\tau}} + \frac{d}{2^{(2-3\alpha)\tau}}\right)^{\bar{\ell}-2} \cdot \sum_{i_2=1}^{d} \sum_{i_1=1}^{d} \Pr\left[(s_1^{(i_1)}, s_2^{(i_2)}) \text{ is a linkable pair}\right] \\
\leq & d\left(1 + \frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}
\end{aligned}$$

□

If $\bar{\ell} \geq 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$, then $d\left(1+\frac{3}{2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}}\right)^{\bar{\ell}-1}$ is exponentially increased. Hence the size of $\bar{\ell}$ is bounded so that $\bar{\ell} < 2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}}$. But, if $\alpha \neq \frac{1}{3}$, then $3\alpha, (2-3\alpha) < 1$ and so $2^{\min\{3\alpha\tau,(2-3\alpha)\tau\}} < \min\left\{d, \frac{\lceil \log N \rceil}{12\tau}\right\}$. It causes the limitation of the size of universe since $|\mathcal{U}| \leq 2^{3\alpha\tau\ell}$ in the case $\alpha \neq \frac{1}{3}$. From these reasons, we use the size of $\mathcal{U}$ to $\{0,1\}^{\tau\ell}$, i.e., $\alpha = \frac{1}{3}$.

Chapter 4

Application to Private Set Union

Previous private set union protocols [KS05, Fri07] based on polynomial representation with additive homomorphic encryption run in linear in the number of players or colluding players because of the following two issues: First, additive homomorphic encryption does not support the polynomial multiplication between encrypted polynomials, hence it requires $O(n)$ rounds to obtain an encryption of a product of polynomials representing sets for the number of players n. Second, as mentioned before, there is no method to uniquely factorize a polynomial defined over message spaces of additive homomorphic encryption schemes, so that they hire a mix-net protocol to recover the resulting set at the final phase.

Recently, Seo et al. [SCK12] overcome this obstacle to present a novel way which represents a set S as a rational function $\frac{1}{f_S} := \frac{1}{\prod_{s_j \in S}(x - s_j)}$ whose poles are elements of the set S using reversed Laurent series, instead of a polynomial. It makes us to resolve the first issue since one can obtain the result corresponded to a set union from the following rational function

additions,

$$\frac{1}{f_{S_1}} + \cdots + \frac{1}{f_{S_n}} = \frac{\gcd(f_{S_1}, \ldots, f_{S_n}) \cdot u}{f_{S_1} \cdot \ \cdots \ \cdot f_{S_n}} = \frac{u}{\mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})}$$

for some polynomial u, which are supported by additive homomorphic encryption schemes.

However, it is still problematic to recover a set from the resulting polynomial without exploiting a mix-net protocol. To resolve this issue, the authors in [SCK12] avoid the use of additive homomorphic encryption schemes and hire secret sharing techniques so that polynomials corresponded to sets are defined over $\mathbb{Z}_q$ for some public prime q. As a result, their protocols bring about the necessity for an honest majority assumption in the malicious case.

In this chapter, we provide efficient constant-round private set union protocols without an honest majority assumption. To do this, we borrow a rational function representation from [SCK12] and generalize it to work on $\mathbb{Z}_\sigma$ which is a message space of Naccache-Stern encryption scheme. Then we complete to construct a protocol by combining it with our second polynomial representation and hiring Naccache-Stern encryption scheme. Our construction have comparable complexities with the previous best result [SCK12] for the honest-but-curious case and are more efficient than previous best result [SCK12] in the malicious model without an honest majority assumption.

Remark that we also provide a constant-round private multi-set union protocol by modifying our encoding function ι in the second polynomial representation so that the same elements have different encoding values. Our protocol is a little bit slower than a previous best result [HKK$^+$11]. However, the public key size of the utilized additive homomorphic encryption in our protocol is $O(1)$, while that of previous work is $O(d)$ for the size d of the multi-set union.

Outline of This Chapter

Section 4.1 reviews a rational function representation with reversed Laurent series over $\mathbb{Z}_q$ for a prime q and $\mathbb{Z}_\sigma$ for a composite σ. Section 4.2 and Section 4.3 provide our private set union protocols for the honest-but-curious case and the malicious case, respectively. We conclude this chapter with extending our set union protocol to a multi-set union protocol in Section 4.4.

4.1 Transforming Our Representation into Rational Function using Reversed Laurent Series

To construct constant round private set union protocols, we adopt rational function representations presented in [SCK12]. In the rational function representation, each player $\mathcal{P}_i$ represents his own set S_i of elements as a rational function $\frac{1}{f_{S_i}}$ where $f_{S_i} := \prod_{s_j \in S_i} (x - s_j)$. It gives the relation

$$\frac{r_1}{f_{S_1}} + \frac{r_2}{f_{S_2}} + \cdots + \frac{r_n}{f_{S_n}} = \frac{u}{\mathrm{lcm}(f_{S_1}, f_{S_2}, \ldots, f_{S_n})}$$

for random polynomials r_i's and some polynomial u. Then each player tries to recover $f(x) = \mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})$ from a polynomial $F(x) = \frac{u}{\mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})}$.

To represent a set as a rational function, the authors in [SCK12] exploit a reversed Laurent series. We briefly introduce a reversed Laurent series.

For a positive integer q, a *reversed Laurent series* (RLS) over $\mathbb{Z}_q$ is a singly infinite, formal sum of the form $f(x) = \sum_{i=-\infty}^{m} f[i]x^i$ $(f[m] \neq 0)$ with an integer m and $f[i] \in \mathbb{Z}_q$ for all i. We define the degree of f by m and denoted $\deg f$. For a RLS $f(x)$, given $d_1 \leq d_2 \leq \deg f$, we denote $f(x)_{[d_1,d_2]} = \sum_{i=d_1}^{d_2} f[i]x^i$. For a rational function f/g with $f, g \in \mathbb{Z}_q[x]$ and

Algorithm 1 RationalToRLS(f, g, k)

Input: $f(x), g(x) \in \mathbb{Z}_q[x]$ with $\deg f < \deg g$, an integer $k > \deg g$

Output: k higher-order terms of the RLS representation of a rational function f/g

1: Compute $F(x) := f(x) \cdot x^k$.
2: Use polynomial division to compute $Q(x)$ and $R(x)$ with $F(x) = g(x) \cdot Q(x) + R(x)$ and $\deg R < \deg g$.
3: Output $Q(x) \cdot x^{-k}$.

$g \neq 0$, we define *the RLS representation of a rational function* f/g by a reversed Laurent series of f/g.

When q is a prime, the RLS representation has the following property:

- The RLS representation for a given rational function is unique.
- Let f, g be polynomials in $\mathbb{Z}_q[x]$ with $\deg f < \deg g$ and $g \neq 0$. Then there exists an algorithm [SCK12] to compute $k(> \deg g)$ high-order terms of the RLS representation of f/g. We describe this algorithm in Algorithm 1. We denote the output of RationalToRLS algorithm which takes polynomials f, g and integer k by RationalToRLS(f, g, k).
- When $2k$ high-order terms of the RLS representation of a rational function f/g such that $f, g \in \mathbb{Z}_q[x]$, $g \neq 0$, and $\deg f < \deg g \leq k$ is given, there exists an efficient algorithm [Sho09, Section 17.5.1] to recover two polynomials $v(x), u(x)$ in $\mathbb{Z}_q[x]$ such that $\frac{v}{u} = \frac{f}{g}$ in $\mathbb{Z}_q(x)$ and $\gcd(v, u) = 1$.

In our protocol, we will represent each player's set S_i as our polynomial representation $f_{S_i} := \prod_{s_j \in S_i}(x - \iota(s_j)) \in \mathbb{Z}_\sigma[x]$ with our encoding function ι, where $\mathbb{Z}_\sigma$ is a message space of Naccache-Stern encryption with $\sigma = \prod_{j=1}^{\bar{\ell}} q_j$

for primes q_j's. Then we convert a rational function of $1/f_{S_i}$ to its RLS over $\mathbb{Z}_\sigma$. Since $\mathbb{Z}_\sigma$ is not a Euclidean domain, one may doubt whether the RationalToRLS algorithm works on $\mathbb{Z}_\sigma[x]$. However, in our protocol, since the conversion requires polynomial divisions only by monic polynomials, the RationalToRLS algorithm works well on $\mathbb{Z}_\sigma[x]$.

After the end of interactions among players in our protocol, each player obtains the $2nk$ higher-order term of the RLS representation of a rational function $\frac{u(x)}{U(x)}$ where $U(x) = \mathrm{lcm}(f_{S_1}(x), \ldots, f_{S_n}(x))$. There is no algorithm to recover $u'(x)$ and $U'(x)$ in $\mathbb{Z}_\sigma[x]$ such that $\frac{u(x)}{U(x)} = \frac{u'(x)}{U'(x)}$. However, from our second polynomial representation, it only requires $U'(x) \bmod q_j$ for each j, and we can obtain $U'(x) \bmod q_j$ from the RLS representation modulo q_j by running polynomial recovering algorithm on $\mathbb{Z}_{q_j}[x]$'s.

The following lemma guarantees that, in a polynomial ring $\mathbb{Z}_\sigma[x]$, a modular operation by a prime divisor q of σ and RationalToRLS algorithm are commutative. This will be utilized to prove the correctness of our protocol.

Lemma 4.1.1. *Let f, g be polynomials in $\mathbb{Z}_\sigma[x]$ with $\deg f < \deg g$ and $g \neq 0$. Suppose that the leading coefficient of g is 1 in $\mathbb{Z}_\sigma$. For each prime q which divides σ and an integer $k > \deg g$,*

$$\mathit{RationalToRLS}(f \bmod q, g \bmod q, k) = \mathit{RatonalToRLS}(f, g, k) \bmod q.$$

Proof. Let RationalToRLS$(f, g, k) = Q(x)x^{-k}$ where $x^k f(x) = Q(x)g(x) + R(x)$ in $\mathbb{Z}_\sigma[x]$ with $R = 0$ or $\deg R < \deg g$. For each polynomial $p(x)$ in $\mathbb{Z}_\sigma[x]$, denote $p(x) \bmod q$ by $p_q(x)$. Then $x^k f_q(x) = Q_q(x)g_q(x) + R_q(x)$ in $\mathbb{Z}_q[x]$, where $R_q = 0$ or $\deg R_q \leq \deg R < \deg g = \deg g_q$. Since the division algorithm uniquely outputs the quotient and the remainder in $\mathbb{Z}_q[x]$, RationalToRLS$(f \bmod q, g \bmod q, k) = Q_q(x)x^{-k} \equiv Q(x)x^{-k} \bmod q$. □

The following lemma gives an information on the distribution of $u_j(x) := u(x) \bmod q_j$ in our protocol which is inevitable to prove the security of our set union protocol. It guarantees that the distribution of $u_j(x)$ and $u(x)$ are uniformly distributed among polynomials in the set of polynomials having degree at most $\deg(\mathrm{lcm}(f_{S_1}, \ldots, f_{S_n})) - 1$ in $\mathbb{Z}_{q_j}[x]$ and $\mathbb{Z}_\sigma[x]$, respectively. The proof of Lemma 4.1.2 is given in [SCK12].

Lemma 4.1.2 ([SCK12, Lemma 1])**.** *Let $f_{S_1}(x), \ldots, f_{S_n}(x) \in \mathbb{Z}_q[x]$ be polynomials of degree $k \geq 1$ for a prime q. Suppose $r_1(x)$, ..., $r_n(x)$ are polynomials in $\mathbb{Z}_q[x]$, chosen uniformly and randomly in the set of polynomials of degree at most $k-1$. Let $u(x)$ be a polynomial such that*

$$\frac{u(x)}{lcm(f_{S_1}(x), \ldots, f_{S_n}(x))} = \sum_{i=1}^{n} \frac{r_i(x)}{f_{S_i}(x)}. \tag{4.1.1}$$

Then u(x) is uniformly distributed among polynomials in the set of polynomials $\in \mathbb{Z}_q[x]$ having degree at most $\deg(lcm(f_{S_1}(x), \ldots, f_{S_n}(x))) - 1$.

4.2 Set Union for Honest-But-Curious Case

Threshold Naccache-Stern Encryption

For a group decryption, it requires a semantically secure, threshold Naccache-Stern additive homomorphic encryption scheme in our protocol. We utilize a threshold version of the Naccache-Stern encryption scheme provided in Section 2.4.

Parameter Setting

Let $\mathcal{U}$ be the universe, n be the number of players, and k be the maximum size of players' datasets. Let d be the maximum size of the set union,

i.e., $d = nk$. Take N so that $\log N > 12 \log |\mathcal{U}|$, which is the modulus of threshold Naccache-Stern additive homomorphic encryption scheme. Put $d_0 = \max\{d, \lceil \log N \rceil\}$ and $\tau = \frac{1}{3}(\log d + 2 \log d_0)$. Set a positive integer ℓ so that $\ell > \log |\mathcal{U}|/\tau$ and a proper positive integer ℓ' so that satisfies the relation (3.2.3). Let $\bar{\ell} = \ell + \ell'$. Note that $\bar{\ell}$ is to be smaller than $\min\left\{d, \frac{\log N}{4 \log \log N}\right\}$. Generate the parameters of the threshold Naccache-Stern encryption scheme, including the size of message space σ, which is a product of $\bar{\ell}$ $(3\tau + 1)$-bit distinct primes q_i's.

Our Set Union Protocol for Honest-But-Curious Case

Our set union protocol against honest-but-curious adversaries is described in Figure 4.1. In our set union protocol, each player computes the higher-order $2nk$ term of the RLS representation of $F_{S_i} = \dfrac{1}{f_{S_i}} = \dfrac{1}{\prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j}))}$ in $\mathbb{Z}_\sigma[x]$ for our encoding function ι and sends its encryption to all others. After interactions among players, each player obtains the high-order $2nk$ terms of the RLS representation of $F(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{1}{f_{S_j}} \cdot r_{i,j} \right)$ in $\mathbb{Z}_\sigma[x]$ for some random polynomials $r_{i,j}$'s chosen by each player $\mathcal{P}_i$. Then each player obtains the high-order $2nk$ terms of the RLS representation of F in $\mathbb{Z}_\sigma[x]$ with group decryption and using this values, recover polynomials $u_j(x)$ and $U_j(x)$ such that

$$\left(\frac{u_j(x)}{U_j(x)} \right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$$

and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$. Thereafter, each player extracts all roots of $U_j(x)$ over $\mathbb{Z}_{q_j}$ for each j and recover all elements based on criteria of our second polynomial representation.

Input: There are $n \geq 2$ honest-but-curious players $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality k. Set $d = nk$. Let $\mathsf{E_{pk}}$ be a threshold Naccache-Stern encryption scheme. Let $\iota : \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$ be our encoding function defined in Section 3.2.

Each player $\mathcal{P}_i$, $i = 1, \ldots, n$:

1. (a) constructs the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i} (x - \iota(s_{i,j}))$ in $\mathbb{Z}_\sigma[x]$ and runs RationalToRLS$(1, f_{S_i}, (2n+1)k - 1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, defines

$$F_{S_i}(x) := \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}.$$

   (b) computes $\tilde{F}_{S_i}$, the encrypted polynomial of F_{S_i} and sends $\tilde{F}_{S_i}$ to all other players.

2. (a) chooses random polynomials $r_{i,j}(x) \in \mathbb{Z}_\sigma[x]$ of degree at most k for all $1 \leq j \leq n$.

   (b) computes the encryption, $\tilde{\phi}_i$, of the polynomial $\phi_i(x) = \sum_{j=1}^{n} F_{S_j} \cdot r_{i,j}$ and sends it to all players.

3. (a) calculates the encryption of the polynomial $F(x) = \sum_{i=1}^{n} \phi_i(x)$.

   (b) performs a group decryption with all other players to obtain the high-order $2nk$ terms of $F(x)$.

4. (a) recovers a polynomial fair of $u_j(x)$ and $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $1 \leq j \leq \bar{\ell}$ such that $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$ and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$, using the high-order $2nk$ terms of $F(x)$ obtained in Step 3 (b).

   (b) extracts all roots of $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all j using a factorization algorithm.

   (c) determines the set union using our rule of representation.

Figure 4.1: Our Set Union Protocol for the Honest-but-Curious Case

Security Analysis

Now, we consider the correctness and privacy of our proposed protocol described in Figure 4.1. The following theorems guarantee the correctness and privacy of our construction in Figure 4.1.

Theorem 4.2.1. *In the protocol described in Figure 4.1, every players learn the set union of private inputs participating players, with high probability.*

Proof. After Step 3 (b), all players obtain the high-order $2nk$ terms of $F(x)$ and hence they obtain the high-order $2nk$ terms of the RLS representation of $F(x) \bmod q_j$ for all $1 \leq j \leq \bar{\ell}$. From these values, using polynomial recovering algorithm, they reconstruct polynomials $u_j(x)$ and $U_j(x)$ such that $\dfrac{u_j(x)}{U_j(x)} \equiv F_j(x) \bmod q_j$, and $\gcd(u_j(x), U_j(x)) = 1$.

At this time, to recover the exact polynomial $\mathrm{lcm}(f_{S_1}, \ldots, f_{S_n}) \bmod q_j$ from the rational function $F_j(x)$, the relation $\gcd(u_j(x), \mathrm{lcm}(f_{S_1}, \ldots, f_{S_n}) \bmod q_j) = 1$ is to be satisfied. In our set union protocol, since $u_j(x)$ is uniformly distributed in $\mathbb{Z}_{q_j}[x]$ from Lemma 4.1.2 and the expected number of roots of a random polynomial is one [Leo06], we may expect that our RLS representation fails to output all the elements in the set union with probability $\frac{d}{q_j} \approx 2^{-2\tau}$, which is the probability that roots of a random polynomial u_j is in the union set. Furthermore, it can be detected if this happens for a certain j, whose probability is about $1 - (1 - \frac{d}{q_j})^{\bar{\ell}} \approx \frac{\bar{\ell} d}{q_j} \leq 2^{-\tau}$. In other words, the probability that each player obtain a wrong polynomial U_j for at least one j is less than $2^{-\tau}$. It is not negligible, but small enough.

Since our polynomial representation can give the exact corresponded set with overwhelming probability, it gives $S_1 \cup \ldots \cup S_n$. □

Theorem 4.2.2. *Assume that the utilized threshold Naccache-Stern additive homomorphic encryption is semantically secure. Then, in our set union pro-*

tocol for the honest-but-curious case described in Figure 4.1, any adversaries $\mathcal{A}$ of colluding fewer than n honest-but-curious players learn no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.

Proof. Since the utilized threshold Naccache-Stern additive homomorphic encryption scheme is semantically secure, each player learns only $F(x) = \sum_{j=1}^{n} (\sum_{i=1}^{n} r_{i,j}(x)) F_{S_j}$ in $\mathbb{Z}_\sigma[x]$. All players contribute to generate the polynomial $(\sum_{i=1}^{n} r_{i,j}(x))$ and the polynomial $\sum_{i=1}^{n} r_{i,j}(x)$ is uniformly distributed and unknown. Moreover, the resulting polynomials $u_j(x)$ are uniformly distributed by Lemma 4.1.2. Hence, no information can be recovered from the polynomial F, U_j's and u_j's, other than that given by revealing the union set. □

Performance Analysis

It is clear that our protocol runs in $O(1)$ rounds. Let us count the computational and communicational costs for each player.

Step 1 (a): requires $\tilde{O}(k)$ multiplications in $\mathbb{Z}_\sigma$ for polynomial expansion of degree k and $O(kd)$ multiplications to compute F_{S_i}.

Step 1 (b): requires $O(d)$ exponentiations for $2d$ encryptions and $O(nd)$ communication costs.

Step 2 (b): requires $O(d^2)$ exponentiations for computing the encryption $\tilde{\phi}_i := \sum_{j=1}^{n} \tilde{F}_{S_j} *_h r_{i,j}$ and $O(nd)$ communication costs.

Step 3 (a): requires $O(nd)$ multiplications for computing $\sum_{i=1}^{n} \tilde{\phi}_i$.

Step 3 (b): requires $O(d)$ exponentiations for decryption share computation for $2d$ ciphertexts and $O(\bar{\ell}\sqrt{dq_i})$ multiplications for solving d DLPs for

$\bar{\ell}$ groups of order q_j's. Note that one has to solve $\bar{\ell}$ DLPs over a group of order q_i for one decryption in Naccache-Stern encryption scheme. In Step 3 (b), one has to solve $2d = 2nk$ DLPs over a group of order q_i for each q_i. It requires $O(\sqrt{dq_i})$ multiplications to solve d DLPs over a group of order q_i [KS01] and hence total complexity of this step is $O(\bar{\ell}\sqrt{dq_j})$ multiplications. The communication cost is $O(nd)$.

Step 4 (a): requires $O(d^2)$ multiplications in $\mathbb{Z}_{q_j}$ to recover $U_j(x)$ using extended Euclidean algorithm for each j.

Step 4 (b): requires $O(d^{1.5+o(1)})$ multiplications in $\mathbb{Z}_{q_j}$ for each j to factor a polynomial of degree d.

Step 4 (c): requires $O(\bar{\ell}d \log d \log q_j)$ bit operations for sorting and $O(d)$ hash computations.

Then the computational complexity is dominated by one of terms $O(d^2)$ exponentiations in Step 2 (b) and $O(\bar{\ell}\sqrt{dq_i})$ multiplications in Step 3 (b). Since one modular exponentiation for a modulus N requires $O(\log N)$ multiplications and $\bar{\ell} < \min\left\{d, \frac{\log N}{4 \log\log N}\right\}$, the computational complexity for each player is $O(d^2) = O(n^2k^2)$ exponentiations in $\mathbb{Z}_N$ and the total complexity is $O(n^3k^2)$ exponentiations in $\mathbb{Z}_N$. The total communication cost for our protocol is $O(n^2d) = O(n^3k)$ $(\log N)$-bit elements.

For the malicious case, we can obtain the set union protocol using general zero-knowledge proof techniques in [KS05] and [SCK12]. It increases the computational and communicational costs. We deal with the malicious case in the next section.

In Table 4.1, we compare communicational and computational complexities with those of previous private set union protocols [KS05, Fri07, SCK12].

We can confirm that our protocols in the honest-but-curious case have comparable complexity with previous best result.

Table 4.1: Comparison with Previous Set-Union Protocols

HBC	Rounds	Communication	Computation	# of Honest Party
[KS05]	$O(n)$	$O(n^3k\tau_N)$	$O(n^4k^2\tau_N\rho_N)$	≥ 1
[Fri07]	$O(n)$	$O(n^2k\tau_N)$	$O(n^2k^2\tau_N\rho_N)$	≥ 1
[SCK12]	$O(1)$	$O(n^4k^2\tau_{p'})$	$O(n^5k^2\rho_{p'})$	$\geq n/2$
Ours	$O(1)$	$O(n^3k\tau_N)$	$O(n^3k^2\tau_N\rho_N)$	≥ 1
Malicious	Rounds	Communication	Computation	# of Honest Party
[Fri07]	$O(n)$	$O((n^2k^2+n^3k)\tau_N)$	$O(n^2k^2\tau_N\rho_N)$	≥ 1
[SCK12]	$O(1)$	$O(n^4k^2\tau_p)$	$O(n^5k^2\tau_p\rho_p)$	$\geq n/2$
Ours	$O(1)$	$O(n^3k^2\tau_N)$	$O(n^3k^2\tau_N\rho_N)$	≥ 1

n: the number of players, k: the maximum size of sets

$\tau_N, \tau_{p'}, \tau_p$: the size of modulus N for Paillier encryption scheme or Naccache-Stern encryption scheme, the size p' of representing domain, the order p of a cyclic group for Pedersen commitment scheme, respectively

$\rho_N, \rho_{p'}, \rho_p$: modular multiplication cost of modulus N for Paillier encryption scheme or Naccache-Stern encryption scheme, p' for the size of representing domain, p for the order of a cyclic group for Pedersen commitment scheme, respectively

4.3 Set Union for Malicious Case

Now, we provide a constant-round set union protocol for the malicious case by exploiting zero-knowledge proof techniques and commitment schemes.

Zero-knowledge Proofs

We exploit the following zero-knowledge proofs for the malicious adversary model. We can efficiently construct the above zero-knowledge proofs for the Naccache-Stern encryption scheme by applying some standard techniques [CS97, CDN01]. We briefly introduce how to construct the following zero-knowledge proofs.

- $\mathsf{ZKPK}[g(x)|\mathcal{E}_{\mathsf{pk}}(g(x)), \mathcal{E}_{\mathsf{pk}}(f(x)), \mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))]$
  This is a zero-knowledge proof that $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ is an encryption of $f(x)g(x)$ when polynomial encryptions $\mathcal{E}_{\mathsf{pk}}(g(x))$, $\mathcal{E}_{\mathsf{pk}}(f(x))$ and $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ are given. In this case, the prover knows only $g(x)$, not $f(x)$. We obtain this protocol by generalizing zero-knowledge proof of correct multiplication which proves $\mathsf{E}_{\mathsf{pk}}(c)$ is an encryption of ab, when $\mathsf{E}_{\mathsf{pk}}(a)$ and $\mathsf{E}_{\mathsf{pk}}(b)$ are given for an additive homomorphic encryption scheme E_{pk}. This protocol requires $O(nk^2)$ exponentiations for computation and $O(nk^2)$ $(\log N)$-bit elements for communication when $f(x)$ is a polynomial of degree $2nk$ and $g(x)$ is a polynomial of degree k.

- $\mathsf{ZKPK}[f(x), g(x)]$
  This is a zero-knowledge proof that $g(x)$ is the RLS representation of $1/f(x)$ when encryptions of $f(x)$ and $g(x)$ are given. From the Lemma 2 in [SCK12], if $f(x)$ and $g(x)$ satisfy the following relation $\deg\left(f(x)g(x) - x^{(\deg f + \deg g)}\right) < \deg f$, then $g(x)$ is the RLS representation of a rational function $1/f(x)$. Hence. it is enough to prove that

the higher-order $\deg(g(x)) + 1$ coefficient of $f(x)g(x)$ are equal to 1, 0, ..., 0. To prove this, the prover first gives $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ with zero-knowledge proof $\mathsf{ZKPK}[g(x)|\mathcal{E}_{\mathsf{pk}}(g(x)), \mathcal{E}_{\mathsf{pk}}(f(x)), \mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))]$ (In this case, the prover also knows $f(x)$, but the protocol is the same.) Then, using 3-round zero-knowledge proof protocols to prove that a ciphertext is an encryption of 0 and a ciphertext is an encryption of 1 [Kor07], the prover proves that the encryption of the higher-order $(\deg g) + 1$ of $\mathcal{E}_{\mathsf{pk}}(f(x) \cdot g(x))$ are encryption of $1, 0, \ldots, 0$. It requires $O(nk^2)$ exponentiations and $O(nk^2)$ $(\log N)$-bit elements for communications when $f(x)$ is a polynomial of degree k and $g(x)$ is a polynomial of degree $2nk$.

Commitment Scheme

We also exploit some equivocal commitment schemes [KO04, MY04] so that the simulator in the malicious adversary model can open the envelope to arbitrary value without being detected by the adversary.

Our Set Union Protocol for Malicious Case

We provide our constant-round private set union protocol which is secure against malicious adversaries in Figure 4.2. The parameters are the same with those of protocols for the honest-but-curious adversaries model in Section 4.2.

Input: There are $n \geq 2$ players $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality k. Set $d = nk$. Let $\mathsf{E_{pk}}$ be a threshold Naccache-Stern encryption scheme. Let $\iota : \{0,1\}^{\tau\ell} \to \mathbb{Z}_\sigma$ be our encoding function defined in Section 3.2.We utilize an equivocal commitment scheme and zero-knowledge proofs protocols.

Each player $\mathcal{P}_i$, $i = 1, \ldots, n$:

1. (a) constructs the polynomial $f_{S_i}(x) = \prod_{s_{i,j} \in S_i}(x - \iota(s_{i,j})) \in \mathbb{Z}_\sigma[x]$ and runs RationalToRLS$(1, f_{S_i}, (2n+1)k - 1)$ to obtain $\left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]}$, defines $F_{S_i}(x) = \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$.
   (b) computes $\tilde{f_{S_i}}$, $\tilde{F}_{S_i}$, the encrypted polynomial of f_{S_i}, F_{S_i} and send it to all other players with proofs of $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (c) chooses random polynomials $r_{i,j}(x)$ of degree at most k for all $1 \leq j \leq n$, and sends a commitment of $\Lambda(r_{i,j})$ to all parties, where $\Lambda(r_{i,j}) = \mathcal{E}_{pk}(r_{i,j})$.
2. (a) opens the commitment to $\Lambda(r_{i,j})$.
   (b) verify zero-knowledge proofs $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.
   (c) sets the leading coefficient to a known encryption of 1.
   (d) calculates $\mu_{i,j}$, the encrypted polynomial of $F_{S_j} \times r_{i,j}$ with proofs of correct multiplication $\mathsf{ZKPK}[r_{i,j}|\Lambda(r_{i,j}), \mathcal{E}_{\mathsf{pk}}(F_{S_{i,j}}), \mu_{i,j}]$ and sends them all other players.
3. (a) calculates the encrypted polynomial of $F(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} F_{S_j} \times r_{i,j}$ and verified all attached proofs.
   (b) performs a group decryption with all other players to obtain the high-order $2nk$ terms of $F(x)$.
4. (a) recovers a polynomial fair of $u_j(x)$ and $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all $1 \leq j \leq \bar{\ell}$ such that $\left(\frac{u_j(x)}{U_j(x)}\right)_{[-2nk,-1]} = (F(x) \bmod q_j)_{[k-1,(2n+1)k-2]} \cdot x^{-(2n+1)k+1}$ and $\gcd(u_j(x), U_j(x)) = 1$ in $\mathbb{Z}_{q_j}[x]$, using the high-order $2nk$ terms of $F(x)$ obtained in Step 3 (b).
   (b) extracts all roots of $U_j(x)$ in $\mathbb{Z}_{q_j}[x]$ for all j using a factorization algorithm.
   (c) determines the set union using our rule of representation.

Figure 4.2: Our Set Union Protocol for the Malicious Case

Analysis

This protocol also runs in $O(1)$ round. The complexities are the same with the protocol for Honest-But-Curious adversary model as a big-O notation except those of running zero-knowledge proof protocols. However, to give a zero-knowledge proof of polynomial multiplication and inverse relation, we need to $O(nk^2)$ communication cost and $O(nk^2)$ computational cost. In particular, a zero-knowledge proof protocol $\mathsf{ZKPK}[r_{i,j}|\Lambda(r_{i,j}), \mathcal{E}_{\mathsf{pk}}(F_{S_{i,j}}), \mu_{i,j}]$ has to run for all $1 \leq i \neq j \leq n$, the total communication complexity and computational complexity are $O(n^3k^2)$ $(\log N)$-bit elements and $O(n^3k^2)$ exponentiation and this is the most expensive part in our malicious protocol.

Theorem 4.3.1. *Assume that there exist zero-knowledge proof protocols and commitment schemes described in the protocol. Suppose the utilized threshold Naccache-Stern encryption scheme is semantically secure. Then, in our set union protocol for the malicious case described in Figure 4.2 is secure. In other words, there exists a simulator $\mathcal{S}$ for a player (or a group of players) operating in the ideal model, such that the view of the players in the ideal model is computationally indistinguishable from the view of the honest players and any adversaries $\mathcal{A}$ of colluding players in the real world.*

Proof. Let $\mathcal{S}$ be a simulator in ideal world, communicating with malicious adversaries $\mathcal{A}$ in the real world, who are able to collude among malicious adversaries. We want to show that malicious adversaries $\mathcal{A}$ in the real world cannot distinguish that the simulator $\mathcal{S}$ does not play in the real world.

The simulator $\mathcal{S}$ operates the following steps:

1. For each honest player i, the simulator $\mathcal{S}$

   (a) chooses monic polynomial f_i such that each such polynomial is relatively prime.

(b) chooses polynomials $r_{i,1}, \ldots, r_{i,n}$ and creates encryptions $\Lambda(r_{i,j})$ from them.

2. The simulator $\mathcal{S}$ performs Step 1 (a), (b), (c) of our set union protocol:

   (a) calculates $F_{S_i}(x) = \left(\frac{1}{f_{S_i}(x)}\right)_{[-(2n+1)k+1,-k]} \cdot x^{(2n+1)k-1}$.

   (b) computes $\tilde{f_{S_i}}$, $\tilde{F}_{S_i}$, the encrypted polynomial of f_{S_i}, F_{S_i} and send it to all other players with proofs of $\mathsf{ZKPK}[f_{S_i}, F_{S_i}]$.

   (c) chooses random polynomials $r_{i,j}(x)$ of degree at most k for all $1 \leq j \leq n$, and sends trapdoor commitment of $\Lambda(r_{i,j})$ to all parties, where $\Lambda(r_{i,j}) = \mathcal{E}_{pk}(r_{i,j})$.

   (d) receives from each malicious player $\alpha \in \mathcal{A}$ the followings: $\tilde{f_{S_\alpha}}$, $\tilde{F}_{S_\alpha}$, $\mathsf{ZKPK}[f_{S_\alpha}, F_{S_\alpha}]$ and trapdoor commitment of $\Lambda(r_{\alpha,j})$ for $1 \leq j \leq n$.

3. The simulator $\mathcal{S}$ extracts witness f_{S_α} using soundness property of zero-knowledge proof $\mathsf{ZKPK}[f_{S_\alpha}, F_{S_\alpha}]$ and trapdoor commitments to $\Lambda(r_{\alpha,j})$ to obtain a polynomial F_α and polynomials $r_{\alpha,j}$ for all $\alpha \in \mathcal{A}$.

4. The simulator $\mathcal{S}$ submits all roots to TTP and return the set union.

5. The simulator prepare to reveal the set union:

   (a) computes $U(x) =: \prod_{s_j \in U} (x - \iota(s_j)) \in \mathbb{Z}_\sigma[x]$ and compute the high-order $2nk$ RLS representation $F(x)$ of a rational function $\frac{u(x)}{U(x)}$ over $\mathbb{Z}_\sigma$ for a randomly chosen $u(x)$ such that $\gcd(u_j(x), U_j(x)) = 1$ where $\gcd(u_j(x), U_j(x)) = 1$, $\deg u < \deg U$.

   (b) computes $U_j(x) := \frac{u_j(x)}{F_j(x)} \bmod q_j$ where $U_j(x) := \prod_{s_j \in U} (x - s_j) \bmod q_j$.

(c) chooses a set of polynomial $r_{i,j}$'s such that $F(x) = \sum_{i=1}^{n} F_{S_j}(x)(\sum_{j=1}^{n} r_{i,j})$ in $\mathbb{Z}_\sigma[x]$.

6. The simulator $\mathcal{S}$ follows the rest of the protocol as described and he opens the trapdoor commitment to reveal an appropriate $\Lambda(r_{i,j})$ for the new chosen polynomial $r_{i,j}$. Then the players calculate an encryption of the polynomial U chosen by the simulator $\mathcal{S}$, and then decrypt it and hence learn the union set.

Note that the simulated transcript produced by the simulator $\mathcal{S}$ consists of ciphertexts, commitments and zero-knowledge proofs. Hence, the simulated transcript is indistinguishable from the real protocol's transcript because of the assumptions that the utilized encryption is semantically secure, the utilized zero-knowledge protocol is simulatable, and the utilized commitment scheme is perfect hiding. Hence the colluding players $\mathcal{A}$ cannot distinguish that the simulator $\mathcal{S}$ is in the real world or not. Also all players obtain the correct answer in both the real world and ideal world.

Therefore, our set union protocol is secure for the malicious case. □

4.4 Extension to Multi-set Union Protocol

We can easily extend our set union protocol to a multi-set union protocol by modifying our encoding function ι. Assume that each player $\mathcal{P}_i$ has a multi-set $S_i \subseteq \mathcal{U}$ for the known universe $\mathcal{U} \subseteq \{0,1\}^{\tau\ell}$. Define a function $\eta : \mathcal{U} \to \mathcal{U}'' \subseteq \{0,1\}^{\tau(\ell+\ell'')}$ by $\eta(s) = s||r$ where r is a randomly chosen element in $\{0,1\}^{\tau\ell''}$. Then each player takes part in our set union protocol with a set $\{\eta(s_1), \ldots, \eta(s_k)\}$ as his set instead of $\{s_1, \ldots, s_k\}$. For the same messages s_1 and s_2, if $\eta(s_1)$ is different from $\eta(s_2)$, one can obtain $\eta(s_1)$ and $\eta(s_2)$ as a part of the set union, so the frequency of s_1 in the union can be revealed. Hence, if all outputs of η are distinct, we can learn multi-set union.

Consider the probability that there exist at least two same values among d values of function η. Then this probability is $1-\left(1-\frac{1}{2^{\tau\ell''}}\right)\cdots\left(1-\frac{d-1}{2^{\tau\ell''}}\right) \approx \frac{d^2}{2^{\tau\ell''}}$ and it is less than $2^{-\lambda}$ if $\ell'' > \frac{\lambda + 2\log d - 1}{\log d}$. For example, when $\lambda = 80$ and $d \approx d_0 \approx 2^{10}$, then ℓ'' is about 10.

Both computational and communicational complexities of our multi-set union protocol are the same with those of our set union protocol as big-O notation. However, it is more heavy than the previous best result [HKK+11], which requires $O(n^2k)$ exponentiations in $\mathbb{F}_q$ and $O(n^2k\log q)$ bits where q is the similar size of the size of the universe. However, the public key size of our protocol is $O(1)$ elements, while that of previous result is $O(d)$ elements for the size d of multi-set union since their construction utilized El Gamal encryption schemes defined over an extension field $\mathbb{F}_{p^d}$ of extension degree d.

Chapter 5

Application to Private Set Intersection for Multiple Use in Storage Model

We consider the private set intersection in storage model. The storage model consists of three entities, set owners, storage, recipients: The owners of sets store their datasets in storage with a private manner. When the recipients want to obtain the intersection of a part of stored sets in the storage, the storage provides ciphertexts corresponded to the set intersection to the recipients. Then, each recipients obtain the set intersection by jointly decrypting given ciphertexts.

To obtain a private set intersection protocol for storage model from previous private set intersection protocols, some technical improvements are required:

1. The set owners can store their own sets without the help of other set owners.

2. The stored set can be utilized multiple times for obtain set intersections among various parts of set owners with maintaining privacy.

3. The recipients should be able to recover the result although they only know the universe of the elements.

Our proposed protocol comes from a private set intersection protocol in [KS05]. To resolve the first issue, we provide a new set encryption so that the ciphertext is an encryption of a product of his polynomial and a random polynomial without revealing a multiplied random polynomial, not solely his polynomial.

For the second issue, we employ a slightly modified version of the threshold Paillier encryption scheme. In modified version, to support t-time uses, the key generation algorithm distributes t different Paillier decryption shares $\mathsf{sk}_{1,i}, \cdots, \mathsf{sk}_{t,i}$ to each recipient i such that $\sum_{i=1}^{m} sk_{j,i} = c_j\phi(N)$ for hidden integers c_j's, RSA modulus N, and the number of recipients m. We also provide a novel way to get only $O(t/m)$ decryption shares to enable t computations.

Finally, we resolve the third issue by adopting our first polynomial representation with a bucket allocation technique borrowing from [FNP04].

The performance analysis shows that the proposed protocol is as efficient as the recently proposed protocols [KS05, SS09] in terms of computation and communication cost. The proposed protocol can be a nice tool for third-party data processing application such as privacy-preserving data publishing [FKWY10].

Outline of This Chapter

Section 5.1 introduces our set encryption. In Section 5.2, we provide basic private set intersection protocol for the honest-but-curious case and the

malicious case. In Section 5.3 we consider extensions of basic private set intersection protocols to storage model.

5.1 Our Set Encryption

Now, we provide our set encryption based on Paillier additive homomorphic encryption scheme. Let E be an encryption algorithm of Paillier additive homomorphic encryption whose message space is $\mathbb{Z}_N$ for RSA modulus N. Let $H : \{0,1\}^* \to \mathsf{CtSp}$ be a collision-resistant hash function onto CtSp, a ciphertextspace* of Paillier encryption scheme. Then an encryption $\mathcal{E}$ of a set S whose cardinality k is defined by the following step: For given a set S, we first compute

$$f_S(x) := \prod_{s_j \in S} (x - s_j) \in \mathbb{Z}_N[x]$$

and choose a random string $\mathsf{str} \in \{0,1\}^*$. Then we compute

$$\mathcal{E}(S)[\ell] := \left(\prod_{j+j'=\ell} H(\mathsf{str}||j)^{f_S[j']} \right) \gamma_\ell^N$$

where γ_ℓ's are randomly chosen elements in $\mathbb{Z}_{N^2}$ for $0 \leq \ell \leq 2k$. Then we define an encryption $\mathcal{E}(S)$ of a set S by $\mathcal{E}(S) := (\mathcal{E}(S)[0], \mathcal{E}(S)[1], \ldots, \mathcal{E}(S)[2k])$.

The following lemma shows that our set encryption is an encryption of a product of a polynomial f_S corresponded a set S and a polynomial r randomly chosen in a set of polynomials of degree less than k.

Lemma 5.1.1. *Let $\mathcal{E}(S) = (\mathcal{E}(S)[0], \mathcal{E}(S)[1], \ldots, \mathcal{E}(S)[2k])$ be an encryption of a set S of cardinality k, based on Paillier additive homomorphic encryption*

*The ciphertextspace of Paillier encryption is $\mathbb{Z}_{N^2}^*$, and it is not public since the factorization of N is secret. Instead of a hash function onto $\mathbb{Z}_{N^2}^*$, we will use a hash function onto $\mathbb{Z}_{N^2}$. Assuming that a hash function outputs random values, the probability that the output of hash function is in $\mathbb{Z}_{N^2} \backslash \mathbb{Z}_{N^2}^*$ is $1 - \frac{1}{p} - \frac{1}{q} + \frac{1}{N}$ for prime factors p, q of N.

algorithm E. *Then,* $\mathcal{E}(S)$ *is an encryption of a product of a polynomial* $f_S := \prod_{s_j \in S}(x - s_j) \in \mathbb{Z}_N[x]$ *and some polynomial* r *in* $\mathbb{Z}_N[x]$. *In particular,* $r = \sum_{j=0}^{k} r[j]x^j$ *is a polynomial such that* $r[j] = \mathsf{D}(H(\mathsf{str}||j))$ *for all* j, *where* D *is a Paillier decryption algorithm corresponded to an encryption* E.

Proof. Let $r[j] := \mathsf{D}(H(\mathsf{str}||j))$ for all $0 \le j \le k$. Then, for all $0 \le \ell \le 2k$,

$$\begin{aligned}
\mathsf{D}(\mathcal{E}(S)[\ell]) &= \mathsf{D}\left(\left(\prod_{j+j'=\ell} H(\mathsf{str}_i||j)^{f_S[j']}\right)\gamma_\ell^N\right) \\
&= \sum_{j+j'=\ell} \mathsf{D}\left(H(\mathsf{str}_i||j)^{f_S[j']}\right) \\
&= \sum_{j+j'=\ell} f_S[j']\mathsf{D}\left(H(\mathsf{str}_i||j)\right) \\
&= \sum_{j+j'=\ell} f_S[j']r_i[j] = (f_S \cdot r_i)[\ell]
\end{aligned}$$

from additive homomorphic property of the Paillier encryption scheme and hence $\sum_{\ell=0}^{2k} \mathsf{D}(\mathcal{E}(S)[\ell])x^\ell = f_S \cdot r$ for a polynomial $r = \sum_{j=0}^{k} r[j]x^j$. □

While the set encryption described in Section 2.4 is an encryption of a polynomial $f_S(x) := \prod_{s_j \in S}(x - s_j) \in \mathbb{Z}_N[x]$, our set encryption of a set S can be interpreted as an encryption of a product of a polynomial f_S and some polynomial r from the view of the previous set encryption. Moreover, by Lemma 5.1.1, since each coefficient $r[j]$ in the polynomial r is a decrypted message of a ciphertext $H(\mathsf{str}||j)$, anyone cannot know $r[j]$ before decrypting $H(\mathsf{str}||j)$ from the fact that the utilized Paillier encryption scheme is semantically secure.

For given ciphertexts $\mathcal{E}(S_1), \cdots, \mathcal{E}(S_n)$ of sets $S_1, \ldots, S_n$ of cardinality k, we can easily obtain an encryption of the intersection of $S_1 \ldots, S_n$:

$$\mathcal{E}(\bigcap_{i=1} S_i)[\ell] := \prod_{i=1}^{n} \mathcal{E}(S_i)[\ell] \tag{5.1.1}$$

for $0 \leq \ell \leq 2k$ using additive homomorphic property of Paillier encryption and these ciphertexts corresponds to an encryption of a polynomial $\sum_{i=1}^{n} r_i \cdot f_{S_i}$ as the previous set encryption in Section 2.4, without even knowing polynomials r_i's by the fact that the utilized Paillier encryption scheme is semantically secure.

In the previous work [KS05], to obtain the encryption of $\sum_{i=1}^{n} r_i \cdot f_{S_i}$ so that each player do not know r_i's without the aid of all players, players have to jointly compute $(r_{j\,j} + r_{j\,j+1} + \cdots + r_{j\,j+c})\mathcal{E}(f_{S_j})$, which is equal to $\mathcal{E}((r_{j\,j} + r_{j\,j+1} + \cdots + r_{j\,j+c}) \cdot f_{S_j})$, where c is the maximum number of colluding players, r_{jl} is the contribution of the player l and $\mathcal{E}(f_{S_j})$ is the set encryption of a set S_j, described in Section 2.4. Hence the previous set encryption requires $O(cnk^2)$ exponentiations for obtaining the encryption of $\sum_{i=1}^{n} r_i \cdot f_{S_i}$ from encryptions of $f_{S_1}, \ldots, f_{S_n}$ so that each player do not know r_i's without the aid of all players. However, our set encryption requires only $O(nk)$ multiplications to obtain the encryption of $\sum_{i=1}^{n} r_i \cdot f_{S_i}$ by the Equation (5.1.1).

It requires $O(k^2)$ exponentiations to encrypt a set S of cardinality k and it is heavier than previous set encryption described in Section 2.4. However, as described in the above paragraph, our set encryption is much efficient than the previous set encryption to combine ciphertexts, which is more frequent computation for multiple use of set encryptions.

5.2 Our Basic Private Set Intersection Protocols

Now, we present our basic set intersection protocols. The efficiency of our protocol is comparable with the recently proposed scheme [SS09]. Also, the security of our protocol will be inevitable to prove the security of our extensions to the storage model.

5.2.1 Honest-But-Curious Case

Our Construction

Our basic PPSI protocol for honest-but-curious case is described in Figure 5.1. In our protocol, each player $\mathcal{P}_i$ has his own dataset S_i of cardinality[†] k and computes $f_{S_i}(x) = \prod_{s_j \in S_i} (x - s_j)$. Then he chooses a random string $\mathsf{str} \in \{0,1\}^*$ and computes an encryption $\mathcal{E}(S_i)$ of a set S_i, presented in Section 5.1, and sends them to all other players. Then each player computes $\mathcal{E}(S)[\ell] = \left(\prod_{i=1}^{n} \mathcal{E}(S_i)[\ell]\right)$ for $0 \leq \ell \leq 2k$. All players perform group decryptions to obtain $f_S = \sum_{i=1}^{n} f_{S_i} \cdot r_{S_i}$ for some hidden polynomials r_{S_i}'s. Then each player evaluates $f_S(x)$ at all elements of his dataset.

[†]If the cardinality of S_i is less than k, each player adds some dummy value so that the cardinality of S_i is to be k.

Input: There are $n \geq 2$ honest-but-curious players $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality k. Let (E, D) be a Paillier threshold additive homomorphic encryption and decryption. Let $H : \{0,1\}^* \to \mathsf{CtSp}$ be a collision resistant hash-function where CtSp is a ciphertext space of the encryption E.

Each player $\mathcal{P}_i$, $i = 1, \ldots, n$:

1. constructs $f_{S_i} = \prod_{s_j \in S_i} (x - s_j)$, chooses a random string $\mathsf{str} \in \{0,1\}^*$, and computes an encryption of S_i:

$$\mathcal{E}(S_i)[\ell] := \left(\prod_{j+j'=\ell} H(\mathsf{str}||j)^{f_{S_i}[j']} \right) \gamma_\ell^N$$

for $0 \leq \ell \leq 2k$ where γ_ℓ is a randomly chosen element in $\mathbb{Z}_{N^2}$ and sends str and $\mathcal{E}(S_i)[\ell]$ to all other players.

2. computes

$$\mathcal{E}(S)[\ell] = \left(\prod_{i=1}^{n} \mathcal{E}(S_i)[\ell] \right)$$

for $0 \leq \ell \leq 2k$.

3. performs group decryptions with all other players to decrypt $\mathcal{E}(S)[\ell]$ for $0 \leq \ell \leq 2k$. Then, each player obtains a polynomial f_S.

4. evaluates $f_S(x)$ at all elements of his dataset. If $f_S(a) = 0$, then a is an element in the intersection of sets.

Figure 5.1: Private Set Intersection for the Honest-But-Curios Case

Security Proof

The following theorem guarantees the correctness of our private set intersection protocol described in Figure 5.1.

Theorem 5.2.1. *In our protocol described in Figure 5.1, each player learns the set intersection of all players' private inputs, $S_1 \cap \ldots \cap S_n$ with overwhelming probability.*

Proof. After Step 2, all players obtain a ciphertext $\mathcal{E}(S)$. By Lemma 5.1.1 and additive homomorphic property,

$$\begin{aligned} \mathsf{D}(\mathcal{E}(S)[\ell]) &= \mathsf{D}(\prod_{i=1}^{n} \mathcal{E}(S_i)[\ell]) = \sum_{i=1}^{n} \mathsf{D}(\mathcal{E}(S_i)[\ell]) \\ &= \sum_{i=1}^{n} (f_{S_i} \cdot r_{S_i})[\ell]. \end{aligned}$$

for $0 \leq \ell \leq 2k$ and some polynomials r_{S_i}'s. Hence, all players obtain $f_S = \sum_{i=1}^{n} f_{S_i} \cdot r_{S_i} = \gcd(f_{S_1}, \ldots, f_{S_n}) \cdot u$ for some polynomial u. If $s_j \in S_i$ is an element of the intersection of datasets, it satisfies $f_S(s_j) = 0$. Therefore, each player learns the set intersection $S_1 \cap \ldots \cap S_n$ by evaluating f_S at all elements of his dataset S_i with overwhelming probability. □

The following theorem gives the privacy of our private set intersection protocol.

Theorem 5.2.2. *Our private set intersection protocol is secure against honest-but-curious adversaries in the random oracle model.*

Proof. Since we assume that the utilized additive homomorphic Paillier encryption scheme is semantically secure, adversaries $\mathcal{A}$ obtain no information from ciphertexts. After group decryption, adversaries $\mathcal{A}$ learn $f_S =$

$\sum_{i=1}^{n} f_{S_i} \cdot r_{S_i}$. However, since the additive homomorphic encryption is semantically secure, adversaries $\mathcal{A}$ cannot know any polynomial r_{S_i} from hash values and $f_S / \gcd(f_{S_1}, \ldots, f_{S_n})$ looks to be random in the set of polynomials of degree $(2k - \deg(\gcd(f_{S_1}, \ldots, f_{S_n})))$ by Lemma 2 in [KS05]. Hence, no information about the private inputs of honest players are revealed, except the intersection of their private datasets. □

Performance Analysis

Table 5.1: Complexity Comparison

HBC	Computation	Communication
[KS05]	$O(cnk^2)$	$O(cnk)$
[SS09]	$O(nk^2 + tnk)$	$O(tnk)$
Ours	$O(nk^2)$	$O(n^2k)$
Mal	**Computation**	**Communication**
[KS05]	$O(n^2k^2)$	$O(n^2k^2)$
[SS09]	$O(nk^2 + tnk)$	$O(nk^2)$
Ours	$O(nk^2)$	$O(n^2k^2)$

n: the number of users, k: the maximum size of user's dataset, t: the number of threshold decryptor, c: the number of colluding players

Our private set intersection protocol in Figure 5.1 runs in $O(1)$ rounds. Let us count the communication and computational complexity for each user.

Step 1: requires $O(k)$ multiplications for polynomial expansion of degree k and $O(k^2)$ exponentiation for computing $\mathcal{E}(f_S)[\ell]$ for all ℓ. And it requires $O(k)$ $(\log N)$-bit communications.

Step 2: requires $O(nk)$ multiplications for each users.

Step 3: requires $O(k)$ exponentiations for group decryptions of all ciphertexts $\mathcal{E}(f_S)[\ell]$ and $O(nk)$ $\log(N)$-bit communications.

Step 4: requires $O(k^2)$ multiplications for evaluating $f_S(x)$ at all elements of his dataset using Horner's rule.

Hence the total computational complexity and communication complexity are $O(nk^2)$ exponentiations in $\mathbb{Z}_{N^2}$ and $O(n^2k)$ $(\log N)$-bit communications, respectively.

In Table 5.1, we provide the comparison our protocol with previous results. These are a similar level of those of the previous best result.

5.2.2 Malicious Case

To achieve the security against malicious adversaries, our basic private set intersection protocol hires zero-knowledge proof of knowledge of multi-exponentiations:

- $\pi_{f_{S_i},\mathcal{E}(S_i),\{\gamma_\ell\}_{\ell=0}^{2k}}$: zero-knowledge proofs of knowledge of a polynomial f_S which is utilized to compute $\mathcal{E}(S)$ where $\mathcal{E}(S)[\ell] := \left(\prod_{j+j'=\ell} H_j^{f_S[j']}\right)\gamma_\ell$ for some public values $H_j \in \mathsf{CtSp}$ and $0 \leq \ell \leq 2k$, a ciphertextspace of Paillier additive homomorphic encryption scheme.

We can construct it using standard techniques [CS97] and Bangerter et al. [BCM05] presented the efficient protocol on $\mathbb{Z}_N^*$ under the strong RSA assumption.

We can efficiently construct this using standard techniques [CS97, BCM05]. It requires $O(k^2)$ exponentiations and $O(k^2)$ $O(\log N)$-bit communications for zero-knowledge proof for each polynomial f_{S_i}.

Our basic private set intersection protocol for the malicious case is described in Figure 5.2.

Input: There are $n \geq 2$ players $\mathcal{P}_i$ with a private input set $S_i \subseteq \mathcal{U}$ of cardinality k. Let (E, D) be a Paillier threshold additive homomorphic encryption and decryption. Let $H : \{0,1\}^* \to \mathsf{CtSp}$ be a collision resistant hash-function where CtSp is a ciphertext space of the encryption E.

Each player $\mathcal{P}_i$, $i = 1, \ldots, n$:

1. constructs $f_{S_i} = \prod_{s_j \in S_i} (x - s_j)$, chooses a random string $\mathsf{str} \in \{0,1\}^*$, and computes

$$\mathcal{E}(S_i)[\ell] := \left(\prod_{j+j'=\ell} H(\mathsf{str}||j)^{f_{S_i}[j']} \right) \gamma_\ell^N$$

for $0 \leq \ell \leq 2k$ where γ_ℓ is a randomly chosen element in $\mathbb{Z}_{N^2}$, along with the proof $\pi_{f_{S_i}, \mathcal{E}(S_i), \{\gamma_\ell\}_{\ell=0}^{2k}}$, and sends str, $\mathcal{E}(S_i)[\ell]$'s, and proofs to all other players.

2. verifies all zero-knowledge proofs and computes

$$\mathcal{E}(S)[\ell] = \left(\prod_{i=1}^{n} \mathcal{E}(S_i)[\ell] \right)$$

for $0 \leq \ell \leq 2k$.

3. performs group decryptions with all other players to decrypt $\mathcal{E}(S)[\ell]$ for $0 \leq \ell \leq 2k$. Then, each player obtains a polynomial f_S.

4. evaluates $f_S(x)$ at elements at his dataset. If $f_S(a) = 0$, then a is an element in the intersection of a set.

Figure 5.2: Private Set Intersection for the Malicious Case

Theorem 5.2.3. *Assume that there exist zero-knowledge proof protocols described in the protocol and the utilized threshold additive Paillier homomorphic encryption scheme is semantically secure. Then our private set intersection protocol described in Figure 5.2 is secure for the malicious adversaries in the random oracle model.*

Proof. We prove this theorem to show that there exists a simulator $\mathcal{S}$ for a player (or a group of players) operating in the ideal model, such that the view of the players in the ideal model is computationally indistinguishable from the view of the honest players and any coalition $\mathcal{A}$ of colluding players in the real world.

First, we describe the response to H-oracle queries. When the adversary queries the random oracle H at any string str and j, the simulator responds to these queries by the following algorithm:

1. If the query (str, j) is already stored in the list L as a tuple $(\mathsf{str}||j,\ r_{\mathsf{str}||j},\ \gamma_{\mathsf{str}||j},\ H(\mathsf{str}||j))$, then it returns the stored value $H(\mathsf{str}||j)$.

2. Otherwise, it chooses a random value $r_{\mathsf{str}||j}$ in $\mathbb{Z}_N$ and $\gamma_{\mathsf{str}||j}$ in $\mathbb{Z}_N$, computes $H(\mathsf{str}||j) := g^{r_{\mathsf{str}||j}}\gamma_{\mathsf{str}||j}^N$ and return it. Then it adds the tuple $(\mathsf{str}||j,\ r_{\mathsf{str}||j},\ \gamma_{\mathsf{str}||j},\ H(\mathsf{str}||j))$ to the list L.

Let $\mathcal{S}$ be a simulator in ideal world, communicating with malicious adversaries $\mathcal{A}$ in the real world, who are able to collude among malicious adversaries.

$\mathcal{S}$ operates the following steps:

1. For each honest player i, a simulator $\mathcal{S}$, it chooses monic polynomial f_i such that each such polynomial is relatively prime.

2. The simulator $\mathcal{S}$ performs Step 1 of our private set intersection protocol described in Figure 5.2:

(a) chooses a random string str and issues H-queries to str and j.

(b) computes $\mathcal{E}(S_i)[\ell] := \left(\prod_{j+j'=\ell} H(\mathsf{str}||j)^{f_{S_i}[j']} \right) \gamma_\ell^N$ for all ℓ and sends them along with a zero-knowledge proof $\pi_{f_{S_i},\mathcal{E}(S_i),\{\gamma_\ell\}_{\ell=0}^{2k}}$.

3. The simulator $\mathcal{S}$ receives from each malicious player $\alpha \in \mathcal{A}$ the following: $\mathcal{E}(S_i)$ and $\pi_{f_{S_i},\mathcal{E}(S_i),\{\gamma_\ell\}_{\ell=0}^{2k}}$.

4. The simulator $\mathcal{S}$ extracts witness f_{S_α} using strong soundness property of zero-knowledge proof $\pi_{f_{S_\alpha},\mathcal{E}(S_\alpha),\{\gamma_\ell\}_{\ell=0}^{2k}}$.

5. The simulator $\mathcal{S}$ obtains all roots of f_{S_α}. Note that a polynomial factorization over $\mathbb{Z}_N$ is impossible. But we can use the techniques proposed in proof of Theorem 16 in [KS05]. Remark that, in our protocol in Section 5.3.2, root finding is possible using Coppersmith algorithm.

6. The simulator $\mathcal{S}$ submits all roots to TTP and returns the set intersection I.

7. The simulator $\mathcal{S}$ prepares to reveal the set intersection to the malicious players $\mathcal{A}$:

   (a) chooses a target polynomial $f_S := \left(\prod_{s \in I} (x - s) \right) \cdot u$ where u is chosen uniformly in the set of polynomials of degree $2k - |I|$.

   (b) chooses a set r_{S_i} for each honest player i such that $f_S = \sum_{i=1}^{n} f_{S_i} \cdot r_{S_i}$.

8. The simulator $\mathcal{S}$ follows the rest of the protocol as described, and responds H-queries to str_i and j so that $\mathsf{D}(H(\mathsf{str}_i||j)) = r_{S_i}[j]$ (i.e., $r_{\mathsf{str}||j} = r_{S_i}[j]$). Then the players calculate the encryption of a polynomial f_S as an encryption of $\mathcal{E}(S)$, and then decrypt it.

Note that the simulated transcript by the simulator consists of a ciphertext or zero-knowledge proofs and hence the simulated transcript is indistinguishable from the real protocol's transcript since the utilized additive Paillier homomorphic encryption is semantically secure and the utilized zero-knowledge proof is simulatable. Hence, the adversary $\mathcal{A}$ cannot distinguish that the simulator $\mathcal{S}$ is in the real world or not. Also all players the correct answer in both the real world and ideal world.

Therefore, our set intersection protocol is secure for the malicious case. □

5.3 Our Set Intersection Protocol in the Storage Model

In this section, we consider two extensions of our set intersection protocols to the storage model. First, we consider the multiple use of our set encryption with private manner. Second, we consider the private set intersection protocol when the recipients do not possesses a superset of the resulting set.

In storage model, we assume that there are at least two recipients and they obtain the intersection of sets by performing group decryptions with all other recipients. We also assume that colluding among all recipients is not allowed.

5.3.1 Set Intersection for Multiple Use

Suppose that set encryptions $\mathcal{E}(S_1), \ldots, \mathcal{E}(S_n)$ are stored in the storage. When the recipients who want to obtain a set intersection of S_i's for all $i \in I \subseteq \{1, \ldots, n\}$, they easily compute a ciphertext $\mathcal{E}(\bigcap_{i \in I} S_i)$ and can obtain

the set intersection $\bigcap_{i \in I} S_i$. However, the reuse of set encryptions reveals some additional information other than the resulting set.

For example, let I_1, I_2 be subsets of $\{1, \ldots, n\}$ and assume that the recipients obtain $\bigcap_{i \in I_1} S_i$ and $\bigcap_{i \in I_2} S_i$ by combining cipertexts and performing group decryption. In these phases, each recipient obtains polynomials $\sum_{i \in I_1} f_{S_i} \cdot r_{S_i}$ and $\sum_{i \in I_2} f_{S_i} \cdot r_{S_i}$. Hence he can induce polynomials $\sum_{i \in I_1} f_{S_i} \cdot r_{S_i} + \sum_{i \in I_2} f_{S_i} \cdot r_{S_i}$ and $\sum_{i \in I_1} f_{S_i} \cdot r_{S_i} - \sum_{i \in I_2} f_{S_i} \cdot r_{S_i}$. The first polynomial is corresponded to the set $(\bigcap_{i \in I_1} S_i) \bigcap (\bigcap_{i \in I_2} S_i)$ and this set is trivially revealed from the resulting sets $\bigcap_{i \in I_1} S_i$ and $\bigcap_{i \in I_2} S_i$. However, the second polynomial is corresponded to the set $\bigcap_{i \in (I_1 \cup I_2) \setminus (I_1 \cap I_2)} S_i$ and this set can not be obtained from only the resulting sets $\bigcap_{i \in I_1} S_i$ and $\bigcap_{i \in I_2} S_i$ if $I_1 \cap I_2$ is not empty. Hence, we have to prevent this information leakage for the reuse of set encryptions.

To resolve this problem, we suggest that the recipients have t different decryption shares and decrypt ciphertexts using different decryption shares at each time for t decryptions. We explain our suggestion by modifying decryption shares of Paillier threshold encryption scheme [FPS01, DJ01]. In setup phase, the dealer generates t different decryption shares $\mathsf{sk}_{j,1}, \ldots, \mathsf{sk}_{j,m}$ such that $\sum_{i=1}^{m} \mathsf{sk}_{j,i} \equiv c_j \phi(N) \bmod N\phi(N)$ for $1 \leq j \leq t$, where c_j's are randomly chosen elements in $\mathbb{Z}_N$ for the RSA modulus N. Then each recipient i uses the decryption share $\mathsf{sk}_{j,i}$ at j-th decryption. It gives a polynomial $c_j(\sum_{i \in I} f_{S_i} r_i)$ corresponding to a ciphertext $\mathcal{E}(\bigcap_{i \in I} S_i)$, not a polynomial $(\sum_{i \in I} f_{S_i} r_{S_i})$. Note that we assume that each recipient uses the same decryption shares for the same ciphertext. Hence, the recipients cannot obtain

$c_{j_1}(\sum_{i\in I} f_{S_i} r_i)$ and $c_{j_2}(\sum_{i\in I} f_{S_i} r_i)$ for the same set I.

To obtain more information except trivial values obtained from the resulting sets, the adversary has to remove any polynomial factors $f_{S_i} \cdot r_i$ in a polynomial $c_j(\sum_{i\in I} f_{S_i} r_i)$, for any $i \in I$. However, since c_j's and leading coefficients of r_i are hidden, one cannot remove any polynomial factors $f_{S_i} \cdot r_i$ by any polynomial operations with high probability. Hence no information except trivial values obtained from the resulting sets, is revealed.

For our suggestion, it requires $O(t)$ to store t decryption shares for each decryptor. We can reduce the required storage to $O(t/m)$ by the following method: Let $h : \{0,1\}^* \to \mathbb{Z}_{N\phi(N)}$ be a hash function. In the setup phase, the dealer generates the decryption shares $\mathsf{sk}_{1,1}, \ldots, \mathsf{sk}_{1,m}$ for the first decryption. Then, the dealer chooses $\mathsf{sk}_{j,(j \bmod m)}$ so that $\mathsf{sk}_{j,(j \bmod m)} + \sum_{i\neq(j \bmod m)} h(\mathsf{sk}_{j,i}) = c_j\phi(N)$ for a randomly chosen value c_j in $\mathbb{Z}_N$. Then with only the decryption shares $\mathsf{sk}_{1,i}$ and $\mathsf{sk}_{(j,i)}$ such that $j \equiv i \bmod m$, each recipient i can generate t decryption shares using hash computations and it requires $O(t/m)$ storage.

5.3.2 Applying Bucket Allocation

Now, we consider the private set intersection protocol, in which the recipient do not possess any dataset except the universe. If the size of universe is similar with the cardinality of each set S_i, the recipient can evaluate at all elements in the universe. However, if the universe is quite larger than a set S_i, it will be a big burden in computational complexity. To overcome this problem, we utilize our first polynomial representation.

However, as mentioned above, the possible size to find a root with Coppersmith algorithm depends on the size of modulus and the degree of a polynomial. More concretely, for given a polynomial $f(x)$ of degree κ defined

over $\mathbb{Z}_N$, the possible size of finding root is less than $\frac{1}{2}N^{\frac{1}{d}}$. Hence, we have to reduce the degree of the resulting polynomial. To do this, we adopt bucket allocation method, introduced in [FNP04]. First, take a Paillier modulus N so that $N \geq (2 \cdot 2^u)^{2.18\kappa}$ for the universe $\mathcal{U} \subseteq \{0,1\}^u$ and a pre-determined small integer κ. Let $\ell = \frac{2k}{\kappa}$ be the number of buckets and $h : \{0,1\}^* \to [\ell]$ be a uniform hash function for the maximum cardinality k of player's datasets. Each set owner arranges his data s_i to belong to a bucket whose index is $h(s_i)$ and stores set encryptions per bucket in the storage. Then the set owner represents his set as a polynomial per each bucket, not as a whole set. Thus, a set is represented by a number of polynomials whose degrees are small and the result of the protocol is also a number of polynomials whose degrees are small, instead one polynomial with large degree.

Chapter 6

Conclusions

We provided two polynomial representations of a set which enable us to uniquely factorize a polynomial satisfying some criteria, defined over $\mathbb{Z}_\sigma$ for a composite integer σ. Our first suggestion works on the message space of Paillier encryption scheme, $\mathbb{Z}_N$ for RSA modulus N, and is obtained by mediating between the sizes of the modulus of Paillier encryption and elements in the domain and then applying Coppersmith small root finding algorithm. Our second suggestion works on the message space of Naccache-Stern encryption scheme, $\mathbb{Z}_\sigma$ for a product σ of small primes, and is obtained by providing our new encoding function.

As applications of our polynomial representation, we provided a private set union protocol. Our constructions are more efficient than the previous best result without an honest majority assumption. We also considered a private set intersection protocol for multiple uses in storage model in which the owners of sets and the recipients are separated.

Bibliography

[BCM05] Endre Bangerter, Jan Camenisch, and Ueli M. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In Serge Vaudenay, editor, *Public Key Cryptography (PKC) 2005*, volume 3386 of *LNCS*, pages 154–171. Springer, 2005.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography (TCC) 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, 2005.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *ACM Symposium on Theory of Computing (STOC) 1988*, pages 1–10. ACM, 1988.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, 2001.

[CKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 213–231. Springer, 2010.

[Cop97] Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report, No. 260, March 1997, 1997.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[CT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *Financial Cryptography (FC) 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, 2010.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography (PKC) 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David

Chaum, editors, *Advances in Cryptology - CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18. Springer, 1984.

[FKWY10] Benjamin C. M. Fung, Rui Chen Ke Wang, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4), 2010.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[FPS01] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In Yair Frankel, editor, *Financial Cryptography (FC) 2000*, volume 1962 of *LNCS*, pages 90–104. Springer, 2001.

[Fri07] Keith B. Frikken. Privacy-preserving set union. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security (ACNS) 2007*, volume 4521 of *LNCS*, pages 237–252. Springer, 2007.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387. Springer, 2001.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *ACM Symposium on Theory of Computing (STOC) 1987*, pages 218–229. ACM, 1987.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume II Basic Applications*. Cambridge University Press, 2004.

[HKK+11] Jeongdae Hong, Jung Woo Kim, Jihye Kim, Kunsoo Park, and Jung Hee Cheon. Constant-round privacy preserving multiset union. IACR Cryptology ePrint Archive, 2011. Available at http://eprint.iacr.org/2011/138.

[Hon12] Hyunsook Hong. Private communication, 2012.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In Omer Reingold, editor, *Theory of Cryptography (TCC) 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, 2009.

[KLC12] Myungsun Kim, Hyung Tae Lee, and Jung Hee Cheon. Mutual private set intersection with linear complexity. In Souhwan Jung and Moti Yung, editors, *Information Security Applications - International Workshop (WISA) 2011*, volume 7115 of *LNCS*, pages 219–231. Springer, 2012.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, 2004.

[Kor07] Michael James Korman. Secure-ballot electronic voting procedures over the internet. Master's thesis, Univdersity of Connecticut, 2007.

[KS01] Fabian Kuhn and René Struik. Random walks revisited: Extensions of pollard's rho algorithm for computing multiple discrete logarithms. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography (SAC) 2001*, volume 2259 of *LNCS*, pages 212–229. Springer, 2001.

[KS05] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology-CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257, 2005.

[Leo06] V. K. Leont'ev. Roots of random polynomials over a finite field. *Matematicheskie Zametki*, 80(2):313–316, 2006.

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 382–400. Springer, 2004.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In Li Gong and Michael K. Reiter, editors, *ACM Conference on Computer and Communications Security (ACM CCS) 1998*, pages 59–66. ACM, 1998.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 308–318. Springer, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in*

Cryptology - EUROCRYPT 1999, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[SCK12] Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. Constant-round multi-party private set union using reversed laurent series. In Marc Fischlin, editor, *Public Key Cryptography (PKC) 2012*, LNCS. Springer, 2012. To appear.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha93] Adi Shamir. On the generation of multivariate polynomials which are hard to factor. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *ACM Symposium on Theory of Computing (STOC) 1993*, pages 796–804. ACM, 1993.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2nd edition, 2009.

[SS09] Yingpeng Sang and Hong Shen. Efficient and secure protocols for privacy-preserving set operations. *ACM Transactions on Information and System Security*, 13(1), 2009.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *Information Security and Cryptography (ICISC) 2006*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.

[Uma08] Christopher Umans. Fast polynomial factorization and modular composition in small characteristic. In Cynthia Dwork, editor, *ACM Symposium on Theory of Computing (STOC) 2008*, pages 481–490. ACM, 2008.

[vzGG99] Joachim von zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 1999.

국문초록

집합의 다항식 표현은 암호화된 다항식 연산을 가능하게 해주는 덧셈을 보존하는 준동형 암호와 결합하여 프라이버시를 보존하는 집합 연산 프로토콜을 설계하는데 널리 이용된다. 그러나 현재 알려진 효율적인 덧셈을 보존하는 준동형 암호의 메시지 공간은 모두 합성수 위수를 갖는 환 $\mathbb{Z}_\sigma$ 뿐이기 때문에 그 위에서 정의된 다항식을 인수분해 할 수 없다는 한계를 지닌다. 이는 다항식 표현을 기반으로 하는 집합 연산 프로토콜에서 마지막 단계에서 얻은 다항식으로부터 대응되는 집합을 어렵게 한다.

본 학위 논문에서는 덧셈을 보존하는 준동형 암호의 메시지 공간 $\mathbb{Z}_\sigma$ 위에서 정의된 특정 조건을 만족하는 다항식을 유일한 방법으로 인수분해 할 수 있는 두 가지 다항식 표현을 제시한다. 첫 번째 방법은 Paillier 암호의 메시지 공간 $\mathbb{Z}_N$(N은 두 소수의 곱)과 결합하여 이용할 수 있는 방법으로 N에 비해 다항식의 근이 상대적으로 작게하여 Coppersmith가 제시하였던 다항식의 작은 근을 찾는 알고리즘을 이용하여 다항식을 인수분해한다.

두번째 방법은 Naccache-Stern 암호의 메시지 공간인 $\mathbb{Z}_\sigma$(σ는 작은 소수들의 곱)과 결합하여 이용할 수 있는 방법이다. $\mathbb{Z}_\sigma[x]$에서는 일반적으로 여러가지 방법의 다항식 인수분해 표현을 갖는다. 이 방법에서는 Naccache-Stern 암호에서 σ의 소인수분해 정보가 알려져 있다는 점에 착안하여 특별한 인코딩 함수를 적용하여 여러 후보들 중에 근이 인코딩 함수의 상에 있는 경우 해당되는 근을 빠른 속도로 찾을 수 있는 방법을 제시한다.

제시한 다항식 표현의 응용으로 상수라운드의 프라이버시를 보존하는 합집합 연산 프로토콜을 제시한다. 덧셈을 보존하는 준동형 암호의 메시지 공간 위에서 정의된 다항식의 인수분해가 불가능하여 기존의 프로토콜에서는 셔플 프로토콜이나 비밀 분산 기법을 사용하여 이를 해결하였다. 이에 상수라운드 안에 프로토콜을 마치지 못하거나 다수의 착한 사람이 필요하다는 가정 하에 상수라운드를 달성하였다. 본 논문에서는 이러한 제

안 없이 기존보다 효율적인 프라이버시를 보존하는 합집합 연산 프로토콜을 제안한다. 또한, 집합 소유자와 결과를 얻고자 하는 자가 다른 상황에서 적용가능한 저장 공간 모델의 프라이버시를 보존하는 교집합 연산 프로토콜에 대해 다룬다.

주요어휘: 다항식 인수분해, 안전한 집합 연산, 덧셈 보존 암호, 저장 공간 모델, 합집합

학번: No. 2008-30086

감사의 글

박사과정에 들어온지 어느 덧 5년이 지나 박사학위를 수여받게 되었습니다. 먼저 한없이 부족한 저를 열과 성을 다하여 지도하여 주신 지도교수님, 천정희 선생님께 감사드립니다. 자연대 학장 업무로도 바쁘신 와중에도 심사위원장으로 참여해주신 김명환 선생님, iSaC의 센터장으로 서울대학교 암호랩을 이끌어 주시며 심사위원으로 참여해주신 이인석 선생님께도 감사드립니다. 프로젝트를 하면서 더 넓은 사고를 할 수 있도록 가르침을 주신 세종대학교 권태경 선생님, 해외 인턴을 경험할 수 있는 기회를 주시고 그 기간 동안 많은 가르침을 주신 NTT의 Okamoto 박사님께도 흔쾌히 심사위원으로 참여해주신점 감사드립니다. 박사과정 중 논문 작업을 하며 많은 것을 가르쳐주신 홍진 선생님과 영남대학교 이윤호 교수님께도 이 자리를 빌어 감사의 말씀을 드립니다.

제가 박사 학위를 수여받기까지 많은 도움을 주신 분들이 떠오릅니다. 함께 암호학을 공부하였던 ISaC과 CHRI의 구성원이었거나 현재 구성원인 많은 선후배님들, 마주칠 때마다 힘이 되는 석사동기들, 바쁘다는 핑계로 모임에 자주 참석하지도 못하는데도 버리지 않고 같이 놀아주는 언제 보든 마음 편한 초, 중, 고, 대학교 친구들, 이 외에도 이런 저런 관계로 저에게 도움을 주신 많은 분들... 지면이 부족하여 이 곳에 일일이 언급하지 못하여 죄송합니다. 직접 찾아뵙고 감사의 마음을 전하도록 하겠습니다.

부족한 큰 아들 박사학위 받기까지 전폭적으로 지원해주신 세상에서 가장 존경하는 아버지, 어머니... 감사하다는 말 만으로는 제 마음을 모두 표현하기 힘들 것 같습니다. 때로는 형 같은 사려깊은 동생 종화와 항상 힘이 되어 주시는 장인어른, 장모님, 처제에게도 고맙다는 말을 전합니다. 마지막으로 언제나 제 옆에서 힘이 되어주는 아내 혜진과 세상이 저희에게 내려준 선물, 귀염둥이 준우에게도 고마움을 전하며 마칩니다.

어제보다 나은 오늘, 오늘보다 나은 내일이 되도록 노력하겠습니다.